

**UTILITY  
PATENT APPLICATION  
TRANSMITTAL**

(Only for new nonprovisional applications under 37 CFR 1.53(b))

Attorney Docket No.

2000 0660A

Total Pages :

First Named Inventor or Application Identifier

Teruto HIROTA et al.

Express Mail Label No.:

**APPLICATION ELEMENTS**

See MPEP chapter 600 concerning utility patent application contents.

ADDRESS TO: Assistant Commissioner for Patents  
Box Patent Application  
Washington, D.C. 20231

1. ☒ Fee Transmittal Form  
(Submit an original, and a duplicate for fee processing)
2. ☒ Specification [Total Pages -176]  
(preferred arrangement set forth below)
- Descriptive title of the Invention
  - Cross References to Related Applications
  - Statement Regarding Fed sponsored R & D
  - Reference to Microfiche Appendix
  - Background of the Invention
  - Brief Summary of the Invention
  - Brief Description of the Drawings (if filed)
  - Detailed Description
  - Claim(s)
  - Abstract of the Disclosure
3. ☒ Drawing(s) (35 USC 113) [Total sheets -71]
4. ☒ Oath or Declaration [Total Pages - 3]
- a.1. ☒ Newly executed (original or copy)
- a.2. ☐ Unexecuted
- b. ☐ Copy from a prior application (37 CFR 1.63(d))  
(for continuation/divisional with Box 17 completed)  
[Note Box 5 below]
- i. ☐ DELETION OF INVENTOR(S)  
Signed statement attached deleting inventor(s)  
named in the prior application, see 37 CFR  
1.63(d)(2) and 1.33(b).
5. ☐ Incorporation By Reference  
(usable if Box 4b is checked)  
The entire disclosure of the prior application, from which  
a copy of the oath or declaration is supplied under Box  
4b, is considered as being part of the disclosure of the  
accompanying application and is hereby incorporated by  
reference therein.

6. Microfiche Computer Program (Appendix)
7. ☐ Nucleotide and/or Amino Acid Sequence Submission  
(if applicable, all necessary)
- a. ☐ Computer Readable Copy
  - b. ☐ Paper Copy (identical to computer copy)
  - c. ☐ Statement verifying identity of above copies

**ACCOMPANYING APPLICATION PARTS**

8. ☒ Assignment Papers (cover sheet & document(s))
9. ☐ 37 CFR 3.73(b) Statement ☐ Power of Attorney  
(when there is an assignee)
10. ☐ English Translation Document (if applicable)
11. ☐ Information Disclosure Statement (IDS)/PTO-1449  
☐ Copies of IDS Citations
12. ☐ Preliminary Amendment
13. ☒ Return Receipt Postcard (MPEP 503)  
(Should be specifically itemized)
14. ☐ Small Entity Statement(s)  
☐ Statement filed in prior application, Status still proper and desired
15. ☒ Certified Copy of Priority Document(s)  
(if foreign priority is claimed)
16. ☐ Other

17. If a CONTINUING APPLICATION, check appropriate box and supply the requisite information:  
☐ Continuation ☐ Divisional ☐ Continuation-in-part (CIP) of prior Application No.

**18. CORRESPONDENCE ADDRESS**

WENDEROTH, LIND & PONACK, L.L.P.  
2033 "K" Street, N.W.  
Suite 800  
Washington, D.C. 20006  
Phone:(202) 721-8200  
Fax:(202) 721-8250

May 26, 2000

05/26/00

3690 U.S. PTO

3690 U.S. PTO

09/580601

05/26/00

TITLE OF THE INVENTION

**A SEMICONDUCTOR MEMORY CARD, PLAYBACK APPARATUS,  
RECORDING APPARATUS, PLAYBACK METHOD, RECORDING METHOD, AND  
COMPUTER-READABLE RECORDING MEDIUM**

5

This application is based on application Nos.  
H11-149893, H11-236724 and H11-372606 filed in Japan, the  
content of which is hereby incorporated by reference.

10 BACKGROUND OF THE INVENTION

1. Field of the Invention

The present invention relates to a semiconductor  
memory card that stores audio data and control data, and  
to a playback apparatus, recording apparatus, playback  
15 method, recording method, and computer-readable recording  
medium relating to such a semiconductor memory card. In  
particular, the present invention relates to improved  
storage of management information and audio data  
distributed as content by a content distribution service,  
20 such as an electronic music distribution service.

2. Description of the Background Art

Recent years have witnessed the gradual introduction  
of the hardware infrastructure necessary for the electronic  
25 distribution of music. This gives rise to the potential  
for great change in the music industry, where products have

been conventionally distributed as packaged software using media such as compact discs (CDs) and cassette tapes.

Electronic music contents (i.e., songs and albums) can be delivered to consumers by having the consumer's personal computer download contents from a server computer operated by a record label. To listen to the downloaded digital music on a portable player, the user needs to store the music data onto a portable recording medium. At present, the most suitable media for storing electronically distributed music data are semiconductor memory cards.

As examples of such, flash ATA cards and COMPACT FLASH cards are already available. Such semiconductor memory cards include a semiconductor device called flash memory (EEPROM - Electrically Erasable Programmable Read-Only Memory). Flash memory is capable of data reads and writes at much higher speeds than MD (MiniDisc) or CD-R (Compact Disc-Recordable). This means that digital music can be transferred in a short time, in spite of its large data size.

As a major disadvantage, semiconductor memory cards carry the risk of allowing users to make illegal copies of copyrighted music that has been downloaded from an electronic music distribution service. Since semiconductor memory cards allow data to be written at higher speeds than CD-R or MD, copying is thought to be a more serious problem for such memory cards. In order

to overcome the potential dangers regarding copyright infringement, digital music has to be encrypted using a secure encryption method before being stored in a semiconductor memory card.

5           One storage method that takes into account the need to prevent unauthorized copying is the title storage method used under DVD-Audio standard. As one example of this method, a "title", which corresponds to a conventional music album, includes a plurality of "contents", which correspond  
10 to tracks on the album. The contents that compose a title are encrypted using an encryption key, called the "title key", chosen by the disc producer before being recorded on a DVD-Audio disc. This title key is encrypted using an encryption key (usually called the "disc key") that is  
15 unique to each DVD-Audio disc and is stored in a sector header region of a DVD-Audio disc. This disc key is itself encrypted using an encryption key (usually called the "master key") chosen by the manufacturers of content decoding apparatuses and is recorded in the lead-in region  
20 of the DVD-Audio disc. The sector header region and lead-in region cannot be accessed by ordinary users, making it extremely difficult for users to illegally obtain the title key recorded on a DVD-Audio disc.

In comparison to magnetic or optical storage media,  
25 semiconductor memory cards have a limited storage capacity, so that it is normally necessary to compress digital music



with a high compression ratio when storing it onto a semiconductor memory card. One encoding method for achieving a sufficiently high compression ratio for digital music is MPEG2-AAC (Motion Pictures Experts Group 2 - Advanced Audio Coding). One characteristic of MPEG2-AAC compression is that it makes use of the limitations of human hearing and so changes the bit length of the data assigned to each audio frame, an audio frame being the smallest playback unit and representing around 20ms of audio. Data with longer bit lengths is assigned to audio frames that have many frequencies within the range of human hearing, while the shorter bit lengths are assigned to audio frames with fewer of such sounds or frequencies outside the range of human hearing.

Since the amount of data assigned to each audio frame in MPEG2-AAC depends on the number of audible frequencies in the frame (or in other words, because MPEG2-AAC uses variable-bitrate (VBR) encoding), high-quality audio contents can be obtained even at high rates of compression. Such audio contents are suited to distribution on a public network and to storage onto semiconductor memory cards that have a limited storage capacity.

### First Problem

When contents are stored according to conventional methods, decoding the title key used to encrypt the music

contents will enable the user to decrypt all of the music contents recorded on a recording medium. This gives rise to the first problem of the exposure of a single title key making it easy for users to decrypt all of the tracks stored on a semiconductor memory card.

While title keys will seldom be exposed, such exposure will result in an immeasurable loss to the copyright holder. With the great advancements in the processing power of home computers in recent years, it is becoming increasingly difficult to say that a title key used to encrypt digital music will be completely safe from decoding. This gives rise to demands for a data construction that will minimize the damage to copyright holders when a title key is exposed.

## **Second Problem**

As copyright protection is necessary for digital music that is to be distributed by electronic music distribution, such music is usually distributed in an encrypted form. Encryption is also required for digital music stored in a semiconductor memory card. However, this gives rise to a second problem that a user who has paid the proper price to purchase digital music will not be able to freely edit the music when it is stored in an encrypted manner on a semiconductor memory card. If the music contents are stored in an encrypted form, it will be very difficult for the user to change the order of tracks or to partially delete

tracks. Considering that the user has paid the proper price,  
it is not desirable to restrict his/her ability to edit  
music contents in this way.

MiniDisc (MD) recorders, which can be used for  
5 recording music in the same way as a semiconductor memory  
card, allow a variety of track editing functions through  
the provision of a TOC (Table of Contents). Such functions  
include the rearranging of the playback order of tracks,  
the division of tracks, and the combining of tracks into  
10 a single track. If semiconductor memory card recorders  
are unable to provide the same functions as conventional  
MD recorders, it is believed that consumers will regard  
semiconductor memory card players as inferior to MD  
recorders, thereby damaging the commercial potential of  
15 semiconductor memory card products.

### **Third Problem**

To provide special playback functions for digital  
music that has been subjected to VBR encoding, as under  
20 MPEG2-AAC, playback apparatuses need to be equipped with  
large-capacity memories. This raises the manufacturing  
cost of such apparatuses, and poses a third problem for  
the background art.

The special playback functions provided by MD or CD  
25 players include the ability to start playback from any track  
on a disc (specifying the playback position), a music search

function that plays back intermittent bursts of music to enable users to skip through tracks forwards or backwards at high speed, and a time search function whereby users can have the playback start from a position inputted as a time measured from the start of the disc. To capture the market currently held by MD or CD players, it is essential for playback apparatuses of semiconductor memory cards to provide the same special playback functions as MD players. When music contents are subjected to constant bitrate (CBR) encoding, playback from a position specified using a time code (such a point one or two minutes from the start of a track) can be performed simply by referring to an address that is offset by an integer multiple of the data size of the unit playback time. However, when music contents are encoded using a VBR method such as MPEG2-AAC, the positions corresponding to one or two minutes ahead of the current position will seldom be offset by an integer multiple of the data size of the unit playback time. As a result, a player will need to refer to a time search table produced in advance to show which addresses correspond to the points one minute and two minutes further ahead.

While a time search table for a short track will not need to include a large number of playback positions, this cannot be said for the time search tables of long tracks, so that the time search tables of long tracks are very large. To provide special playback features, a playback apparatus

has to access the time search table having first loaded it into its memory. Since long tracks have large time search tables, this means that a playback apparatus has to be provided with a large memory for storing the time search table. This also increases the manufacturing costs of playback apparatuses.

#### SUMMARY OF THE INVENTION

It is a first object of the present invention to provide a semiconductor memory card that protects the copyrights of music contents stored therein while allowing users to edit the music contents.

It is a second object of the present invention to provide a playback apparatus that can perform special playback functions such as forward and backward search for music contents recorded on a semiconductor memory card without using a large-capacity memory.

The first object of the present invention can be achieved by a semiconductor memory card that stores at least one audio track, including: a protected area that can be accessed by a device connected to the semiconductor memory card only if the device has been found to be authentic, the protected area storing an encryption key sequence composed of a plurality of encryption keys arranged into a predetermined order; and an unprotected area that can be accessed by any device connected to the semiconductor

memory card, the unprotected area storing at least one audio track and management information, the at least one audio track including a plurality of encrypted audio objects, and the management information showing which encryption  
5 key, out of the plurality of encryption keys, corresponds to each audio object stored in the unprotected area.

With the stated construction, a plurality of audio objects can be encrypted using a plurality of encryption keys, so that should the encryption key used to encrypt  
10 a particular audio object be decoded and exposed, such decoding will only enable that particular audio object to be decrypted and so will have no effect on other audio objects. This means that the present semiconductor memory card minimizes the damage caused by the exposure of one of the  
15 encryption keys.

Here, each audio track may further include (1) attribute information and (2) link information for each audio object included in the audio track, the attribute information showing a type, out of type (a), type (b), type  
20 (c) and type (d), for each audio object, type (a) being an entire audio track, type (b) being a first part of an audio track, type (c) being a middle part of an audio track, and type (d) being an end part of an audio track, and the link information for each audio object that is type (b)  
25 or type (c) showing which audio object follows the audio object.



middle part of an audio track, or (d) corresponds to an end part of an audio track.

When invalid data is present at the start of an audio frame, the length of this invalid data and the length of the valid data in the audio frame can be set in the block information. As a result, when the user records a radio broadcast where the disc jockey talks over the intro of a song, a suitable data offset can be set in the block information to have the song played back without the part of the intro that includes the disc jockey's voice. Such editing operations can be performed by merely indicating what data should not be played back in the block information and are performed with the audio objects in their encrypted state. This means that tracks can be edited at high speed.

The second object of the present invention can be achieved by a recording apparatus for a semiconductor memory card, including: a first generating unit for successively generating audio frames from an input signal received from outside the recording apparatus, an audio frame being a smallest amount of data that can be independently decoded; a writing unit for creating a file on the semiconductor memory card and writing the successively generated audio frames into the file; a second generating unit for generating, whenever the writing unit has written a predetermined number of audio frames into a file, a piece of entry information showing a data length of an audio



element that is composed of the audio frames written into the file, wherein whenever the second generating unit has generated a predetermined number of pieces of entry information, the writing unit creates a new file and writes  
5 the audio frames successively generated thereafter into the new file.

When an audio stream is for a music album which includes a long track, the long track is divided into a plurality of files to ensure that the number of pieces of entry  
10 information for a single file does not exceed a predetermined number. Limiting the number of pieces of entry information in a file suppresses the size of the management information of a file. This management information is used by a playback apparatus as described  
15 below. When a playback apparatus reads a file and commences playback of the audio object included in the file, the playback apparatus also reads the management information for the file and stores it in an internal memory. This management information needs to be kept in the memory as  
20 long as the playback of the audio object continues. When the playback of this audio object ends, the following audio object is read. When playback commences for this following audio object, the corresponding management information is read and overwritten into internal memory of the playback  
25 apparatus to take the place of the management information that was hitherto stored.

09500001.033000  
The playback apparatus therefore repeatedly performs  
a process that loads only the management information for  
the audio object currently being played back into its  
internal memory. This enables playback apparatuses with  
5 limited memory capacity to perform special playback  
functions such as forward and backward search.

The assignment of the plurality of audio objects to  
audio tracks and the order to be used when playing back  
audio tracks is determined by the management information,  
10 so that tracks can be freely edited by merely updating the  
management information.

#### BRIEF DESCRIPTION OF THE DRAWINGS

These and other objects, advantages and features of  
15 the invention will become apparent from the following  
description thereof taken in conjunction with the  
accompanying drawings which illustrate a specific  
embodiment of the invention. In the Drawings:

FIG. 1 shows the appearance of a flash memory card  
20 31 when viewed from above;

FIG. 2 shows the construction of the flash memory  
card 31 when viewed from below;

FIG. 3 shows the hierarchical composition of the flash  
memory card 31 in the embodiments;

25 FIG. 4A shows the special region, the authentication  
region and the user region provided in the physical layer

of the flash memory card 31;

FIG. 4B shows the composition of the authentication region and the user region in the file system layer;

FIG. 5 shows the detailed composition of the file  
5 system layer;

FIG. 6 is a representation of when the AOB file "AOB001.SA1" is divided into five parts that are stored in clusters 003, 004, 005, 00A, and 00C;

FIG. 7 shows one example of the settings of the  
10 directory entries and file allocation table when the AOB file "AOB001.SA1" is recorded in a plurality of clusters;

FIGs. 8A and 8B show what directories are provided in the user region and the authentication region in the file system layer when the above two types of data are  
15 recorded in the application layer, as well as what kind of files are recorded in which directories;

FIG. 9 shows the correspondence between the file "AOBSA1.KEY" and the AOB files in the SD\_Audio directories;

FIG. 10 shows the hierarchical composition of the  
20 data in an AOB file;

FIG. 11A shows the parameters stipulated by ISO/IEC 13818-7 standard in tabular form;

FIG. 11B shows the parameters that should be used when encoding a file in MPEG-Layer 3 (MP3) format in tabular  
25 form;

FIG. 11C shows the parameters that should be used

when encoding a file in Windows Media Audio (WMA) format  
in tabular form;

FIG. 12 shows the detailed construction of an  
AOB\_FRAME;

5        FIG. 13 shows how the byte length of the audio data  
in each of three AOB\_FRAMEs is set;

FIG. 14 shows the correspondence between the  
sampling\_frequency and the number of AOB\_FRAMEs included  
in an AOB\_ELEMENT;

10       FIG. 15 shows examples of the playback periods of  
AOB\_ELEMENTs and the playback periods of AOB\_FRAMEs;

FIG. 16 shows what is reproduced when the AOBs and  
AOB\_BLOCKs recorded in an AOB file are consecutively played  
back;

15       FIG. 17 shows the hierarchical composition of the  
PlaylistManager and TrackManager used in the embodiments  
in detail;

FIG. 18 shows the sizes of the PlaylistManager and  
the TrackManager;

20       FIG. 19 shows the correspondence between the TKIs  
shown in FIG. 17 and the AOBs and AOB files shown in FIG.  
16;

FIG. 20 shows the detailed data composition of the  
TKTMSRT shown in FIG. 17;

25       FIG. 21 shows one example of the TKTMSRT;

FIG. 22 shows the detailed composition of the TKGI;

009250" 052600

FIGs. 23A and 23B show the composition of the BIT;

FIG. 23C shows the Time\_Length field;

FIG. 24 shows cluster 007 to 00E into which the AOB composed of AOB\_ELEMENT#1 to AOB\_ELEMENT#4 are stored;

5        FIG. 25 shows how the next AOB\_FRAME#x+1 to be played back is set when forward search is performed starting from the AOB\_FRAME#x in an arbitrary AOB\_ELEMENT#y in an AOB;

10        FIGs. 26A and 26B shows how an AOB, an AOB\_ELEMENT, and an AOB\_FRAME that correspond to an arbitrary playback time code are specified;

FIGS. 27A and 27B show the deletion of a track;

FIG. 28A shows the TrackManager after the deletion of a track has been performed several times;

15        FIG. 28B shows how a new TKI and AOB file are written when "Unused" TKIs are present in the TrackManager;

FIGS. 29A and 29B show the TKIs are set when two tracks are combined to produce a new track;

FIG. 30A shows a Type1 AOB;

FIG. 30B shows Type2 AOBs;

20        FIG. 31A shows the combining of a plurality of tracks into a single track for a combination of a Type1+ Type2+ Type2+ Type1 AOB;

25        FIG. 31B shows the combining of a plurality of tracks into a single track for a combination of a Type1+ Type2+ Type2+ Type2+ Type1 AOB;

FIG. 32A shows a pattern where a Type1 AOB is present

at the end of a preceding track and a Type1 AOB is present at the start of a next track;

FIG. 32B shows a pattern where a Type1 AOB is present at the end of a first track and a Type2 AOB is present at the start of a next track;

FIG. 32C shows a pattern where a Type1 and Type2 AOB are present at the end of a first track and a Type1 AOB is present at the start of a next track;

FIG. 32D shows a pattern where a Type1 and Type2 AOB are present at the end of a first track and a Type2 and a Type1 AOB is present at the start of a next track;

FIG. 32E shows a pattern where two Type2 AOBs are present at the end of a first track and a Type1 is present at the start of a next track;

FIGS. 33A and 33B show the division of a track to produce two tracks;

FIGS. 34A and 34B show the content of the SD\_Audio directory entries in the SD\_Audio directory including the AOB file "AOB003.SA1" before and after the division of the track;

FIG. 35A shows the division of an AOB midway through AOB\_ELEMENT#2;

FIG. 35B shows the two AOBs, AOB#1 and AOB#2, obtained by dividing an AOB midway through AOB\_ELEMENT#2;

FIG. 36 shows how the BIT is set when an AOB is divided as shown in FIG. 35;

FIG. 37 shows a specific example of changes in the BIT before and after division;

FIG. 38 shows a specific example of changes in the TKTMSRT before and after division;

5 FIG. 39A shows the format of a DPL\_TK\_SRP;

FIG. 39B shows the format of a PL\_TK\_SRP;

FIG. 40 shows the interrelation between the Default\_Playlist\_Information, the TKIs, and the AOB files;

10 FIG. 41 shows example settings for the Default\_Playlist and several PLIs;

FIG. 42 shows how the DPL\_TK\_SRPs correspond to TKIs using the same notation as FIG. 40;

FIGS. 43A and 43B show how the order of tracks is rearranged;

15 FIGS. 44A and 44B show how the Default\_Playlist, TrackManager, and AOB files will be updated when DPL\_TK\_SRP#2 and TKI#2 are deleted from the Default\_Playlist shown in FIG. 40;

20 FIGS. 45A and 45B show how a new TKI and DPL\_TK\_SRP are written when an "Unused" TKI and DPL\_TK\_SRP are present; FIGS. 46A and 46B show how tracks are combined;

FIGS. 47A and 47B shows how a track is divided;

25 FIG. 48 shows the appearance of a portable playback apparatus for the flash memory card 31 of the present embodiments;

FIG. 49 shows one example of the display on the LCD

panel when a playlist is selected;

FIGS. 50A to 50E show examples of the display on the LCD panel when a track is selected;

FIGS. 51A to 51C show example operations of the jog  
5 dial;

FIG. 52 shows the internal construction of the reproduction apparatus;

FIG. 53 shows how data is transferred in and out of the double buffer 15;

FIG. 54A and 54B shows how areas in the double buffer  
10 15 are cyclically allocated using ring pointers;

FIG. 55 is a flowchart showing the AOB file read procedure;

FIG. 56 is a flowchart showing the AOB file output  
15 procedure;

FIG. 57 is a flowchart showing the AOB file output procedure;

FIG. 58 is a flowchart showing the AOB file output procedure;

FIGS. 59A to 59D show how the playback time code  
20 displayed in the playback time code frame on the LCD panel  
5 is updated in accordance with the updating of the variable  
Play\_time;

FIG. 60 is a flowchart shows the processing of the  
25 CPU 10 when the forward search function is used;

FIGS. 61A to 61D show how the playback time code is



incremented when the forward search function is used;

FIGS. 62A and 62B show specific examples of how the time search function is used;

FIG. 63 is a flowchart showing the processing in the editing control program;

FIG. 64 is a flowchart showing the processing in the editing control program;

FIG. 65 is a flowchart showing the processing in the editing control program;

FIG. 66 shows one example of a recording apparatus for recording data onto the flash memory card 31;

FIG. 67 shows the hardware configuration of the recording apparatus;

FIG. 68 is a flowchart showing the processing during recording;

FIG. 69 shows the hardware construction of the flash memory card 31;

FIG. 70 shows the communication sequence used when a playback apparatus connected to the flash memory card 31 reads the encryption key FileKey and plays back AOBs; and

FIG. 71 shows the details of the communication sequence used when mutual authentication is performed in FIG. 70.

## DESCRIPTION OF THE PREFERRED EMBODIMENTS

The following describes a semiconductor memory card (flash memory card) that is an embodiment of the present invention, with reference to the attached figures.

5 The following paragraphs are arranged into a hierarchy using reference numbers with the notation given below.

{x1-x2\_x3-x4}

10 The length of a reference number shows the level of the topic in the hierarchy. As a specific example, the number x1 is the number of drawing that is being referred to in the explanation. The drawings attached to this specification have been numbered in the order in which they  
15 are referred to in the specification, so that the order of the drawings roughly matches the order of the explanation. The explanation of certain drawings has been divided into sections, with the reference number x2 giving the section number of a section in the explanation of a drawing indicated  
20 by the reference number x1. The reference number x3 shows the number of an additional drawing that is provided to show the details of the section indicated by the section number x2. Finally, the reference number x4 shows the number of a section in the explanation of this additional  
25 drawing.

## FIRST EMBODIMENT

### {1-1\_2} External Appearance of the Flash Memory Card 31

The present explanation starts with the external appearance of the flash memory card 31. FIG. 1 shows the appearance of the flash memory card 31 when viewed from above, while FIG. 2 shows the construction of the flash memory card 31 when viewed from below. As shown in FIGS. 1 and 2, the flash memory card 31 is around the same size as a postage stamp, and so is large enough to be held by hand. Its approximate dimensions are 32.0mm long, 24.0mm wide, and 2.0mm thick.

The flash memory card 31 can be seen to have nine connectors on its bottom edge for connecting the card to a compatible device and a protect switch 32 on one side to enable the user to set whether overwriting of the stored content of the flash memory card 31 is permitted or prohibited.

### {3-1} Physical Construction of the Flash Memory Card 31

FIG. 3 shows the hierarchical structure of the semiconductor memory card (hereafter referred to as the "flashmemory card 31") of the present embodiment. As shown in FIG. 3, the flash memory card 31 is constructed with a physical layer, a file system layer and an application layer in the same way as a DVD (Digital Video Disc), though the logical and physical constructions of these layers are

very different to those on a DVD.

### **{3-2} Physical Layer of the Flash Memory Card 31**

The following describes the physical layer of the  
5 flash memory card 31. The flash memory is composed of a  
plurality of sectors, each of which stores 512 bytes of  
digital data. As one example, a 64MB flash memory card  
31 will have a storage capacity of 67,108,864  
(=64\*1,024\*1,024) bytes, so that this card will include  
10 131,072(=67108864/512) valid sectors. Once the number of  
replacement sectors, which are provided for use in case  
of errors, is subtracted, the remaining number of valid  
sectors into which various kinds of data can be written  
is around 128,000.

#### **{3-2\_4A-1} Three Regions in the Physical Layer**

The three regions shown in FIG. 4A are provided in  
the storage area composed of these valid sectors. These  
regions are the "special region", the "authentication  
20 region" and the "user region", and are described in detail  
below. The user region is characterized in that a device  
to which the flash memory card 31 is connected can freely  
read or write various kinds of data from or into this region.  
Areas within the user region are managed by a file system.

25 The special region stores a media ID that is a value  
uniquely assigned to each flash memory card 31. Unlike

the user region, this region is read-only, so that the media ID stored in the special region cannot be changed.

The authentication region is a writeable region, like the user region. This region differs from the user region in that a device connected to the flash memory card 31 can access (i.e., read or write data in) the authentication region only if the flash memory card 31 and the device have first confirmed that each other is an authentic device. In other words, data can only be read from or written into the authentication region if mutual authentication has been successfully performed by the flash memory card 31 and the device connected to the flash memory card 31.

### **{3-2\_4A-2} Uses of the Three Regions**

#### **in the Physical Layer**

When the device connected to the flash memory card 31 writes data into the flash memory card 31, the region used to store this data will depend on whether copyright protection is necessary for the data being written. When data that requires copyright protection is written into the flash memory card 31, the data is encrypted using a predetermined encryption key (called a "FileKey") before being written into the user area. This FileKey can be freely set by the copyright holder and, while the use of this FileKey provides some level of copyright protection, the FileKey

used for encrypting the written data is itself encrypted to make the copyright protection more secure. Any value obtained by subjecting the media ID stored in the special region into a predetermined calculation can be used to encrypt the FileKey. The encrypted FileKey produced in this way is stored in the authentication region.

Since data that requires copyright protection is subjected to a two-step encryption process where the data is encrypted using a FileKey that is itself encrypted based on the media ID, copyright infringement, such as the production of unauthorized copies of this data, will be extremely difficult.

### **{3-2\_4B-1} Overview of the File System**

As can be understood, the construction of the physical layer of the flash memory card 31 strengthens the copyright protection of the data written in the flash memory card 31. The following describes the file system layer present on this physical layer. While the file system layer of a DVD uses a UDF (Universal Disk Format)-type file system, the file system layer of the flash memory card 31 uses a FAT (File Allocation Table) -type file system, as described in ISO/IEC 9293.

FIG. 4B shows the construction of the authentication region and the user region in the file system layer. As shown in FIG. 4B, the authentication region and the user

region in the file system each include "partition boot sectors", a "file allocation table (FAT)", a "root directory", and a "data region", meaning that the authentication region and the user region have the same construction. FIG. 5

shows the various parts of these file systems in more detail. The following describes the construction of the user region with reference to FIGS. 4A, 4B and 5.

#### **{3-2\_4B-2} Partition Boot Sectors**

The partition boot sectors are sectors that store the data that will be referred to by a standard personal computer that is connected to the flash memory card 31 when the flash memory card 31 is set as the boot disk for the operating system (OS) of the personal computer.

#### **{3-2\_4B-3\_5} Data Region**

The data region can be accessed by a device connected to the flash memory card 31 in units no smaller than a "cluster". While each sector in the flash memory card 31 is 512 bytes in size, the cluster size is 16KB, so that the file system layer reads and writes data in units of 32 sectors.

The reason the cluster size is set at 16KB is that when data is written onto the flash memory card 31, part of the data stored in the flash memory card 31 first has

to be erased before the write can be performed.

The smallest amount of data that can be erased in the flash memory card 31 is 16KB, so that setting the smallest erasable size as the cluster size means that data writes can be favorably performed. The arrow ff2 drawn using a broken line in FIG. 5 shows the plurality of clusters 002,003,004,005 . . . included in the data region. The numbers 002,003,004,005,006,007,008 . . . used in FIG. 5 are the three-digit hexadecimal cluster numbers that are exclusively assigned to identify each cluster. Since the smallest unit by which access can be performed is one cluster, storage positions within the data region are indicated using cluster numbers.

### **{3-2\_4B-4\_5} File Allocation System**

The file allocation system has a file system construction in accordance with ISO/IEC 9293 standard, and so is made up of a plurality of FAT values. Each FAT value corresponds to a cluster and shows which cluster should be read after the cluster corresponding to the FAT value. The arrow ff1 shown by a broken line in FIG. 5 shows the plurality of FAT values 002,003,004,005 . . . that are included in the file allocation table. The numbers 002,003,004,005 . . . assigned to each FAT value show which cluster corresponds to each FAT value and therefore are the cluster numbers of the clusters corresponding to



the FAT values.

### **{3-2\_4B-5\_5-1} Root Directory Entries**

The "root directory entries" are information showing  
5 what kinds of files are present in the root directory. As  
specific examples, the "filename" of an existing file, its  
"filename extension", the "revision time/date" and "number  
of first cluster in file" showing where the start of the  
file is stored can be written as the root directory entry  
10 of a file.

### **{3-2\_4B-5\_5-2} Directory Entries for Subdirectories**

Information relating to files in the root directory  
is written as root directory entries, though information  
15 relating to subdirectories is not written as the root  
directory entries. Directory entries for subdirectories  
are instead produced in the data region. In FIG. 5, the  
SD-Audio directory entry given in the data region is one  
example of a directory entry for a subdirectory. Like a  
20 root directory entry, an SD-Audio directory entry includes  
the "filename" of a file present in this subdirectory, its  
"filename extension", the "revision time/date" and "number  
of first cluster in file" showing where the start of the  
file is stored.

25

### **{3-2\_4B-5\_6-1} Storage Format for AOB Files**

The following describes the file storage method by showing how a file named "AOB001.SA1" is stored in the SD-Audio directory, with reference to FIG. 6. Since the smallest unit by which the data region can be accessed is one cluster, the file "AOB001.SA1" needs to be stored in the data region in parts that are no smaller than one cluster. The file "AOB001.SA1" is therefore stored having first been divided into clusters. In FIG. 6, the file "AOB001.SA1" is divided into five parts in keeping with the cluster size, and the resulting parts are stored into the clusters numbered 003, 004, 005, 00A, and 00C.

### **{3-2\_4B-5\_7-1} Storage Format for AOB Files**

When the file "AOB001.SA1" is divided up into parts and stored, a directory entry and the file allocation table need to be set as shown in FIG. 7. FIG. 7 shows one example of how the directory entry and file allocation table need to be set when the file "AOB001.SA1" is stored having been divided up into parts and stored. In FIG. 7, the start of the file "AOB001.SA1" is stored in cluster 003, so that cluster number 003 is written into "the number of first cluster in file" in the SD-Audio directory entry to indicate the cluster storing the first part of the file. As shown in FIG. 7, the following parts of the file "AOB001.SA1" are stored in clusters 004 and 005. As a result, while

the FAT value 003(004) corresponds to cluster 003 that stores the first part of the file "AOB001.SA1", this value indicates cluster 004 as the cluster storing the next part of the file "AOB001.SA1". In the same way, while the FAT values 004(005) and 005(00A) respectively correspond to clusters 004 and 005 that store the next parts of the file "AOB001.SA1", these values respectively indicate cluster 005 and cluster 00A as the clusters storing the next parts of the file "AOB001.SA1". By reading the clusters with the cluster numbers written into these FAT values in order as shown by the arrows fk1, fk2, fk3, fk4, fk5 . . . in FIG. 7, all of the parts produced by dividing the file "AOB001.SA1" can be read. As explained above, the data region of the flash memory card 31 is accessed in units of clusters, each of which is associated with a FAT value. Note that the FAT value that corresponds to the cluster storing the final part of an AOB file (the cluster 00C in the example shown in FIG. 7) is set the cluster number FFF to show that the corresponding cluster stores the final part of a file.

This completes the explanation of the file system in the flash memory card 31 of the present invention. The following describes the application layer that exists on this file system.

### **{3-3} Overview of the Application Layer**

#### **in the Flash Memory Card 31**

An overview of the application layer in the flash memory card 31 is shown in FIG. 3. As shown by the arrow PN2 drawn with a broken line in FIG. 3, the application layer in the flash memory card 31 is composed of presentation data and navigation data that is used to control the playback of the presentation data. As shown by the arrow PN2, the presentation data includes sets of audio objects (AOB sets) that are produced by encoding audio data that represents music, for example. The navigation data includes a "PlaylistManager" (PLMG) and a "TrackManager" (TKMG).

#### **{3-3\_8A,B-1} Directory Composition**

FIGS. 8A and 8B show what kind of directories are present in the user region and the authentication region in the file system layer when these two types of data are stored in the application layer, as well as showing what files are arranged into these directories.

The filenames "SD\_AUDIO.PLM" and "SD\_AUDIO.TKM" in FIG. 8A indicate the files in which the PlaylistManager (PLMG) and TrackManager (TKMG) composing the navigation information are stored. Meanwhile, the filenames "AOB001.SA1", "AOB002.SA1", "AOB003.SA1", "AOB004.SA1", . . . indicate the files ("AOB" files) storing the audio objects that are the presentation data.

The letters "SA" in the filename extension of the filename "AOB0xx.SA1" are an abbreviation for "Secure Audio", and show that the stored content of this file requires copyright protection. Note that while only eight AOB files are shown in the example in FIG. 8A, a maximum of 999 AOB files can be stored in an SD-Audio directory.

When copyright protection is required for presentation data, a subdirectory called an "SD-Audio directory" is provided in the authentication region and an encryption key storing file "AOBSA1.KEY" is produced in this SD-Audio directory.

FIG. 8B shows the encryption key storing file "AOBSA1.KEY" that is stored under the "SD-Audio" legend (i.e., within the "SD-Audio directory"). This encryption key storing file "AOBSA1.KEY" stores a sequence of encryption keys that is produced by arranging a plurality of encryption keys into a predetermined order.

The SD-Audio directory shown in FIGS. 8A and 8B is stored in a server computer managed by a record label that uses electronic music distribution. When a consumer orders a music content, the corresponding SD-Audio directory is compressed, encrypted and transmitted to the consumer via a public network. The consumer's computer receives this SD-Audio directory, decrypts it, decompresses it and so obtains the original SD-Audio directory. Note that the expression "public network" here refers to any kind of

network that can be used by the public, such as a wired communication network, e.g., an ISDN network, or a wireless communication network, e.g., a mobile telephone system. It is also possible for a consumer's computer to download  
5 an AOB file from a server computer operated by a record label and then produce an SD-Audio directory, such as that shown in FIGS. 8A and 8B, in the flash memory card 31.

### **{3-3\_9-1} Correspondence between the**

#### **"AOBSA1.KEY" file and the AOB files**

FIG. 9 shows the correspondence between the "AOBSA1.KEY" file in the SD-Audio directory and the AOB files. The FileKeys used when encrypting files in the user region shown in FIG. 9 are stored in the corresponding  
15 encryption key storing file in the authentication region.

The encrypted AOB files and the encryption key storing file correspond according to the predetermined rules (1), (2), and (3) described below.

(1) The encryption key storing file is arranged into  
20 a directory with the same directory name as the directory in which the encrypted file is stored. In FIG. 9, AOB files are arranged into the SD-Audio directory in the user region and the encryption key storing file is arranged into a directory called the SD-Audio directory in the  
25 authentication region, in accordance with this rule.

(2) The encryption key storing file is given a filename



file "AOB003.SA1" corresponds to the FileKey whose storage position is indicated by the "FileKey Entry#3". As can be understood from rule (3), different FileKeys are used to encrypt different AOB files, with these FileKeys being  
5 stored in "FileKey Entries" with the serial numbers "001", "002", "003", "004" etc., given in the filenames of the corresponding AOB files.

Since each AOB file is encrypted using a different FileKey, the exposure of the encryption key used for one  
10 AOB file will not enable users to decrypt other AOB files. This means that when AOB files are stored in an encrypted form on a flash memory card 31, the damage caused by the exposure of one FileKey can be minimized.

### 15 {3-3\_10-1} Internal Composition of an AOB file

The following describes the internal composition of an AOB file. FIG. 10 shows the hierarchical data structure of an AOB file. The first level in FIG. 10 shows the AOB file, while the second level shows the audio object (AOB)  
20 itself. The third level shows the AOB\_BLOCKS, the fourth level an AOB\_ELEMENT, and the fifth level an AOB\_FRAME.

The AOB\_FRAME on the fifth level in FIG. 10 is the smallest unit composing the AOB, and is composed of audio data in ADTS (Audio Data Transport Stream) format and an  
25 ADTS header. Audio data in ADTS format is encrypted according to MPEG2-AAC (Low Complexity Profile) format and



is stream data that can be played back at a transfer rate of 16Kbps to 144Kbps. Note that the transfer rate for PCM (Pulse Code Modulation) that is recorded on a conventional compact disc is 1.5Mbps, so that data in ADTS format  
5 generally uses a lower transfer rate than PCM. The data construction of a sequence of AOB\_FRAMEs is the same as the sequence of audio frames included in an audio data transport stream distributed by an electronic music distribution service. This means that the audio data  
10 transport stream to be stored as AOB\_FRAME sequence is encoded according to MPEG2-ACC standard, encrypted, and transmitted on a public network to the consumer. AOB files are produced by dividing the transmitted audio data transport stream into a sequence of AOB\_FRAMEs and storing  
15 these AOB\_FRAMEs.

### **{3-3\_10-1\_11} MPEG2-AAC**

MPEG2-AAC is described in detail in ISO/IEC 13818-7:1997(E) "Information Technology - Generic Coding  
20 of Moving Pictures and Associated Audio Information - Part 7 Advanced Audio Coding (AAC)".

It should be noted that audio objects can only be compressed according to MPEG2-AAC using the parameters in the parameter table shown in FIG. 11A that is defined in  
25 ISO/IEC13818-7. This parameter table is composed of "Parameter" column, a "Value" column, and a "Comment"

column.

The legend "profile" in the Parameter column shows the only LC-profile can be used, as stipulated under ISO/IEC 13838-7. The legend "sampling\_frequency#index" in the Parameter column shows that the sampling frequencies "48kHz, 44.1kHz, 32kHz, 24kHz, 22.05kHz, and 16kHz" can be used.

The legend "number\_of\_data\_block\_in\_frame" in the Parameter column shows that the ratio of one header to one raw\_data\_block is used.

Note that while this explanation describes the case where AOB\_FRAMEs are encoded according to MPEG-AAC format, AOB\_FRAMEs may instead be encoded according to another format, such as MPEG-Layer3 (MP3) format or Windows Media Audio (WMA). When doing so, the parameters shown in the parameter tables of FIG. 11B or FIG. 11C must be used.

### **{3-3\_10-2\_12} Composition of an AOB\_FRAME**

While each AOB\_FRAME includes audio data that is encoded according to the restrictions described above, the data length of the audio data in each AOB\_FRAME is restricted to a playback time of only 20ms. However, since MPEG2-AAC is a variable bitrate (VBR) encoding method, the data length of the audio data in each AOB\_FRAME will vary. The following describes the composition of an AOB\_FRAME, with reference to FIG. 12.

The first level in FIG. 12 shows the overall



the data length x1 will be written in the ADTS header of AOB\_FRAME#1, the data length x2 will be written in the ADTS header of AOB\_FRAME#2, and the data length x3 will be written in the ADTS header of AOB\_FRAME#3.

5           Although the audio data is encrypted, the ADTS header is not, so that a playback device can know the data length of the audio data in an AOB\_FRAME by reading the data length given in the ADTS header of the AOB\_FRAME.

          This completes the explanation of an AOB\_FRAME.

#### 10           **{3-3\_10-4} AOB\_ELEMENT**

          The following describes the AOB\_ELEMENT shown on the fourth level in FIG. 10.

          An "AOB\_ELEMENT" is a group of consecutive AOB\_FRAMES.

15          The number of AOB\_FRAMES in an AOB\_ELEMENT depends on the value set as the sampling\_frequency\_index shown in FIG. 11A and the encoding method used. The number of AOB\_FRAMES in an AOB\_ELEMENT is set so that the total playback time of the included AOB\_FRAMES will be around two seconds, with  
20          this number depending on the sampling frequency and encoding method used.

#### **{3-3\_10-5\_14} Number of AOB\_FRAMES in an AOB\_ELEMENT**

          FIG. 14 shows the correspondence between the sampling  
25          frequency and the number of AOB\_FRAMES included in an AOB\_ELEMENT. The number N given in FIG. 14 represents the



15 shows a plurality of AOB\_BLOCKS, while the second level shows a plurality of AOB\_ELEMENTS. The third level shows a plurality of AOB\_FRAMES.

As shown in FIG. 15, an AOB\_ELEMENT has a playback period of around 2.0 seconds, while an AOB\_FRAME has a playback period of 20 milliseconds. The "TMSRT\_entry" given to each AOB\_ELEMENT shows that the data length of each AOB\_ELEMENT is given in the time search table. By referring to the TMSRT\_entries, a playback apparatus can perform a forward or backward search where, for example, intermittent bursts of music are played back by repeatedly playing back 240 milliseconds of audio data and then skipping two seconds of audio data in the desired direction.

#### **{3-3\_10-7} AOB\_BLOCK**

This completes the explanation of an AOB\_ELEMENT. The following describes the concept of the AOB\_BLOCKS shown on the third level of the data construction of an AOB file given in FIG. 10.

Each "AOB\_BLOCK" is composed of valid AOB\_ELEMENTS. Only one AOB\_BLOCK exists in each AOB\_FILE. While an AOB\_ELEMENT has a playback period of around two seconds, an AOB\_BLOCK has a maximum playback period of 8.4 minutes. The 8.4 minute limitation is imposed to restrict the size of the time search table to 504 bytes or less.

### **{3-3\_10-8} Restriction of the Time Search Table**

The following describes in detail why the size of the time search table is restricted by limiting the playback period.

5           When a playback apparatus performs a forward or backward search, the playback apparatus skips the reading of two seconds of audio data before playing back 240 milliseconds. When skipping two seconds of data, the playback apparatus could in theory refer to the data lengths shown in the ADTS headers of AOB\_FRAMES, though this would mean that the playback apparatus would have to consecutively detect 100 (2 seconds/20 milliseconds) AOB\_FRAMES just to skip two seconds of audio data. This would amount to an excessive processing load for the playback apparatus.

10           To reduce the processing load of a playback apparatus, the read addresses for data at two-second intervals can be written into a time search table that is then referred to by the playback apparatus when performing a forward or backward search. By writing information that enables read addresses that are two or four seconds ahead or behind to be found quickly into the time search table (such information being the data sizes of AOB\_ELEMENTS), a playback apparatus will only need to refer to this information when performing a forward or backward search.

25           The data size of audio data with a playback period of two seconds will depend on the bitrate used when playing back





AOB is present in each AOB file.

The invalid areas are regions that are read and written along with the AOB\_BLOCKS and are stored in the same clusters as the AOB\_BLOCKS. The start and end position of the AOB\_BLOCKS within an AOB are shown by BITs included in the navigation data. These BITs are described in detail later in this specification.

This completes the explanation of what data is stored in an AOB file. The following describes what kind of content is played back when the eight AOBs and AOB\_BLOCKS shown in the AOB file in FIG. 9 are successively read.

### **{3-3\_10-10\_16}**

FIG. 16 shows the playback content when the AOBs and AOB\_BLOCKS in this AOB file are successively read. The first level in FIG. 16 shows the eight AOB files in the user region, while the second level shows the eight AOBs recorded in these AOB files. The third level shows the eight AOB\_BLOCKS included in these AOBs.

The fifth level shows the titles of five contents composed by these AOB files. In this example, the "contents" are the five songs SongA, SongB, SongC, SongD, and SongE, while the "title" is a music album composed of these five songs. The broken lines AS1, AS1, AS3, . . . AS7, and AS8 show the correspondence between the AOB\_BLOCKS and the parts into which the album is divided, so that the

fourth level in FIG. 16 shows the units used to divide the music album shown on the fifth level.

By referring to the broken lines, it can be seen that the AOB\_BLOCK included in AOB#1 is a song (SongA) with a playback period of 6.1 minutes. The AOB\_BLOCK included in AOB#2 is a song (SongB) with a playback period of 3.3 minutes. The AOB\_BLOCK included in AOB#3 is a song (SongC) with a playback period of 5.5 minutes. In this way, "AOB001.SA1" to "AOB003.SA1" each correspond to a different song. The sixth level of FIG. 16 is a track sequence composed of tracks TrackA to TrackE. These tracks TrackA-TrackE correspond to the five songs SongA, SongB, SongC, SongD, and SongE, and are each treated as a separate playback unit.

On the other hand, AOB#4 has a playback period of 8.4 minutes and is the first (or "head") part of the song SongD that has a playback period of 30.6 minutes. The AOB\_BLOCKS included in AOB#5 and AOB#6 are middle parts of the song SongD and also have playback periods of 8.4 minutes. The AOB\_BLOCK included in AOB#7 is the end part of the song SongD and has a playback period of 5.4 minutes. In this way, a song that has a total playback period of 30.6 minutes is divided into (8.4 + 8.4 + 8.4 + 5.4-minute) parts that are each included in a different AOB. As can be seen from FIG. 16, every song included in an AOB file is subjected to a maximum playback period of 8.4 minutes.

This explanation clearly shows that limiting the



In this specification, the term "track" refers to a meaningful playback unit for users, so that when copyrighted music is stored on a flash memory card 31, each song is a separate track. Conversely, when an "audio book" (i.e., copyrighted literature stored as recorded audio) is recorded on a flash memory card 31, each chapter or paragraph can be set as a separate track. The TrackManager is provided to manage a plurality of AOBs recorded in a plurality of AOB files as a group of tracks.

A Playlist sets the playback order of a plurality of tracks. A plurality of Playlists can be included in the PlaylistManager.

The following describes the TrackManager with reference to the drawings.

#### **{17-1\_18} Detailed Composition of the PlaylistManager and TrackManager**

FIG. 17 shows the detailed composition of the PlaylistManager and TrackManager in this embodiment as a hierarchy. FIG. 18 shows the sizes of the PlaylistManager and the TrackManager. The right side of FIG. 17 shows the items on the left side in more detail, with the broken lines indicating which items are being shown in more detail.

As shown in FIG. 17, the TrackManager is composed of the Track Information (TKI) #1, #2, #3, #4 . . . #n, as shown by the broken line h1. These TKIs are information

for managing the AOBs recorded in AOB files as tracks, and each correspond to a different AOB file. From FIG. 17, it can be seen that each TKI is composed of

Track\_General\_Information (TKGI),

- 5 Track\_Text\_Information (TKXTI\_DA) in which text information exclusive to a track can be written, and a Track\_Time\_Search\_Table (TKTMSRT) that serves as a time search table.

10 From FIG. 18, it can be seen that each TKI has a fixed size of 1,024 bytes, which means that total size of the TKGI and the TKXTI\_DA is fixed at 512 bytes due to the size of the TKTMSRT being fixed at 512 bytes. In the TrackManager, a total of 999 TKIs can be set.

15 As shown by the broken line h3, the TKTMSRT is composed of a TMSRT\_Header and TMSRT\_entries #1, #2, #3 . . . #n.

#### **{17-2\_19} Correspondence of TKI with AOB files and AOBs**

FIG. 19 shows how the TKIs shown in FIG. 17 correspond to the AOB files and AOBs shown in FIG. 16. The boxes on  
20 the first level in FIG. 19 show a sequence of tracks composed of tracks TrackA to TrackE, the large frame on the second level shows the TrackManager, while the third and fourth levels show the eight AOB files given in FIG. 16. The eight AOB files are recorded in the eight AOBs shown in FIG. 16,  
25 and compose a music album including tracks TrackA, TrackB, TrackC, TrackD, and TrackE. The second level shows the

eight TKIs. The numbers "1", "2", "3", "4" assigned to each TKI are the serial numbers used to identify each TKI, with each TKI corresponding to the AOB file that has been given the same serial number 001,002,003,004,005 . . . .

5        With this in mind, it can be seen from FIG. 19 that TKI#1 corresponds to the file "AOB001.SA1", that TKI#2 corresponds to the file "AOB002.SA1", TKI#3 corresponds to the file "AOB003.SA1", and TKI#4 corresponds to the file "AOB004.SA1". The correspondence between TKIs and  
10    AOB\_FRAMES is shown by the arrows TA1, TA2, TA3, TA4 . . . in FIG. 19.

In this way, each TKI corresponds to a different AOB recorded in an AOB file and gives detailed information that applies only to the corresponding AOB.

#### 15    {17-3\_20} Data Composition of a TKTMSRT

The following describes the information that applies to single AOBs recorded in AOB files, starting with the TKTMSRT. FIG. 20 shows the data composition of the TKTMSRT  
20    in detail.

The right side of FIG. 20 shows the detailed data composition of the time search table header (TMSRT\_Header). In FIG. 20, the TMSRT\_Header has a data size of eight bytes, and is made up of three fields. The first two bytes are  
25    a TMSRT\_ID, the next two bytes are reserved, and the final four bytes are a Total TMSRT\_entry\_Number.

A unique ID for identifying the TMSRT is recorded in the "TMSRT\_ID". The total number of TMSRT\_entries in the present TMSRT is recorded in the "Total TMSRT\_entry Number".

5

### {17-3\_21-1} Specific Example of the TKTMSRT

The following describes a TKTMSRT in detail. FIG. 21 shows one example of a TKTMSRT. The left side of FIG. 21 shows an AOB, while the right side shows the corresponding TKTMSRT. The AOB on the left side of FIG. 21 is composed of a plurality of AOB\_ELEMENTS numbered #1, #2, #3 . . . #n that occupy the regions numbered AR1, AR2, AR3 . . . ARn to the right.

15 The numbers such as "0", "32000", "64200", "97000", "1203400", and "1240000" show the relative addresses of areas AR1, AR2, AR3, ARn-1, ARn occupied by the AOB\_ELEMENTS with respect to the start of the AOB\_BLOCK. As examples, AOB\_ELEMENT#2 is recorded at a position that is at a distance  
20 "32000" from the start of the AOB\_BLOCK, while AOB\_ELEMENT#3 is recorded at a position that is at a distance "64200" from the start of the AOB\_BLOCK and AOB\_ELEMENT#n-1 is recorded at a position that is at a distance "1203400" from the start of the AOB\_BLOCK.

25 It should be noted that the distance between each occupied region and the start of the AOB\_BLOCK is not a

multiple of a certain value, meaning that the regions occupied by AOB\_ELEMENTS are not of the same size. The reason the occupied regions have different sizes is that the varying amounts of data are used to encode each AOB\_FRAME.

5        Since the size of the region occupied by each AOB\_ELEMENT differs, it is necessary to inform a playback apparatus in advance of the position of each AOB\_ELEMENT in an AOB when performing a jump to the start of an AOB\_ELEMENT. For this purpose, a plurality of TMSRT\_entries are given  
10    in the TKTMSRT. The arrows RT1, RT2, RT3 . . . RTn-1, RTn show the correspondence between the regions AR1, AR2, AR3 . . . ARn-1, ARn occupied by each AOB\_ELEMENT and TMSRT\_entry#1, TMSRT\_entry#2, TMSRT\_entry#3 . . . TMSRT\_entry#n-1, TMSRT\_entry#n. In other words, the size  
15    of the region AR1 occupied by AOB\_ELEMENT#1 is written in the TMSRT\_entry#1, while the sizes of the regions AR2 and AR3 occupied by AOB\_ELEMENT#2 and AOB\_ELEMENT#3 are written in the TMSRT\_entries #2 and #3.

      Since the occupied area AR1 takes up the region from  
20    the start of the AOB to the start of the AOB\_ELEMENT#2 "32000", the size "32000" (=32000-0) is written in the TMSRT\_entry#1. The occupied area AR2 takes up the region from the start of the AOB\_ELEMENT#2 "32000" to the start of the AOB\_ELEMENT#3 "64200", so that the size "32200"  
25    (=64200-32000) is written in the TMSRT\_entry#2. The occupied area AR3 takes up the region from the start of



the AOB\_ELEMENT#3 "64200" to the start of the AOB\_ELEMENT#4  
"97000", so that the size "32800" (=97000-64200) is written  
in the TMSRT\_entry#3. In the same way, the occupied area  
ARn-1 takes up the region from the start of the  
5 AOB\_ELEMENT#n-1 "1203400" to the start of the AOB\_ELEMENT#n  
"1240000", the size "36600" (=1240000-1203400) is written  
in the TMSRT\_entry#n-1.

#### **{17-3\_21-2} How the TKTMSRT is read**

10 In this way, the data sizes of AOB\_ELEMENTS are written  
in a time search table. However, since the data length  
of each AOB\_BLOCK is restricted to a maximum of 8.4 minutes,  
the total number of AOB\_ELEMENTS included in a single AOB  
is limited to a predetermined number ("252" as shown in  
15 FIG. 20) or less. Since the number of AOB\_ELEMENTS is  
restricted, the number of TMSRT\_entries corresponding to  
AOB\_ELEMENTS is also restricted, which restricts the size  
of the TKTMSRT including these TMSRT\_entries to within a  
predetermined size. Since the size of the TKTMSRT is  
20 restricted, a playback apparatus can read and use TKIs in  
the following way.

The playback apparatus reads a certain AOB and on  
commencing playback of the AOB, reads the corresponding  
TKI and stores it in a memory. This corresponding TKI  
25 is kept in the memory while the playback of this AOB continues.  
Once the playback of the AOB ends, the following AOB is

read, and when the playback of this AOB commences, the playback apparatus overwrites the TKI corresponding to this following AOB into the memory in place of the old TKI. This next TKI is kept in the memory while the playback of this following AOB continues.

By reading and storing TKIs in this way, the necessary capacity of the memory in the playback apparatus can be minimized while still enabling special playback functions such as forward and backward search to be realized. While the present embodiment describes the case where the data length from the first address of an AOB\_ELEMENT to the first address of the next AOB\_ELEMENT is written in the TMSRT\_entry, relative addresses from the start of the AOB\_BLOCK to the first addresses of AOB\_ELEMENTS may be written in there instead.

### **{17-3 21-3} Specifying a Cluster**

Including an AOB ELEMENT

The following describes how an AOB\_ELEMENT may be read using the TKTMSRT. The TKTMSRT includes the size of each AOB\_ELEMENT, so that when AOB\_ELEMENT#y, which is the  $y^{\text{th}}$  AOB\_ELEMENT from the start of an AOB, is to be read, the cluster u that satisfies Equation 1 given below is calculated, and data positioned with the offset v from the start of the cluster u is read.

### Equation 1

Cluster  $u = (\text{Total of the TMSRT\_entries from AOB\_ELEMENT\#1 to AOB\_ELEMENT\#y-1} + \text{DATA\_Offset}) / \text{Cluster size}$

5    Offset  $v = (\text{Total of the TMSRT\_entries from AOB\_ELEMENT\#1 to AOB\_ELEMENT\#y-1} + \text{DATA\_Offset}) \bmod \text{Cluster size}$

      where  $c = a \bmod b$  indicates that  $c$  is the remainder produced when  $a$  is divided by  $b$

10       The DATA\_Offset is written in the BIT and is described later in this specification.

### {17-4}   TKTXI\_DA

      This completes the explanation of the time search  
15   table (TKTMSRT). The following describes the Track\_Text\_InformationDataArea (TKTXI\_DA) recorded in the upper part of the TKTMSRT.

      The Track\_Text\_Information Data Area (TKTXI\_DA) is used to store text information showing the artist name,  
20   album name, mixer, producer, and other such information. This area is provided even when such text information does not exist.

### {17-5}   TKGI

25       The following describes the TKGI recorded in the upper part of the TKTXI\_DA. In FIG. 17, several sets of

information shown as the identifier "TKI\_ID" of the TKI,  
the TKI number "TKIN", the TKI size "TKI\_SZ", a link pointer  
to the next TKI "TKI\_LNK\_PTR", block attributes  
"TKI\_BLK\_ATR", a playback period "TKI\_PB\_TM", the audio  
5 attributes "TKI\_AOB\_ATR", an "ISRC", and block information  
"BIT". Note that only some of this information has been  
shown in FIG. 17 to simplify the representation.

#### **{17-5\_22-1} TKGI**

10 The following describes the composition of a TKGI  
in detail, with reference to FIG. 22. The difference  
between FIG. 17 and FIG. 22 is that the data composition  
of the TKGI that was shown in FIG. 17 is arranged on the  
left side of this drawing, and that the bit compositions  
15 of "TKI\_BLK\_ATR", "TKI\_AOB\_ATR" and "ISRC" are clearly  
shown.

#### **{17-5\_22-2} TKI\_ID**

A unique ID for a TKI is written in the "TKI\_ID".  
20 In the present embodiment, a two-byte "A4" code is used.

#### **{17-5\_22-3} TKIN**

A TKI number in the range of 1 to 999 is written in  
the "TKIN". Note that the TKIN of each TKI is unique. In  
25 the present embodiment, the position of each TKI in the  
TrackManager is used as the TKIN. This means that "1" is

written as the TKI number of TKI#1, "2" is written as the TKI number of TKI#2, and "3" is written as the TKI number of TKI#3.

5    **{17-5\_22-4}   TKI\_SZ**

The data size of the TKI in byte units is written in the "TKI\_SZ". In FIG. 22, 1,024 bytes is given as the data size of the TKI so that each TKI in the present embodiment is 1,024 bytes long.

10    **{17-5\_22-5}   TKI\_LNK\_PTR**

The TKIN of the TKI to which the present TKI is linked is written in the "TKI\_LNK\_PTR". The following describes such links between TKIs.

15        When a track is composed of a plurality of AOBs which are recorded in a plurality of AOB files, these AOB files will be managed as a single track by linking the plurality of TKIs that correspond to these AOB files. To link a plurality of TKIs, it is necessary to show the TKI of the  
20    AOB file that follows after the AOB file of the present TKI. Accordingly, the TKIN of the TKI that follows the present TKI is written in TKI\_LNK\_PTR.

**{17-5\_22-6\_19}   TKI\_LNK\_PTR**

25        The following describes the settings made for the TKI\_LNK\_PTR in the eight TKIs shown in FIG. 19. The track

information numbered #1 to #3 and #8 each correspond to separate tracks, so no information is set in their TKI\_LNK\_PTR. The track information TKI#4,TKI#5,TKI#6,TKI#7 correspond to the four AOB files that compose TrackD, so that the next track information is indicated in the TKI\_LNK\_PTR of these TKIs. As shown by the arrows TL4, TL5, and TL6 in FIG. 19, "TKI#5" is set in the TKI\_LNK\_PTR of TKI#4, "TKI#6" is set in the TKI\_LNK\_PTR of TKI#5, and "TKI#7" is set in the TKI\_LNK\_PTR of TKI#6.

As a result, a playback apparatus can refer to the TKI\_LNK\_PTRs given in the TKIs corresponding to these four AOB files and so find out that the four TKIs TKI#4 to TKI#7 and the four AOB files "AOB004.SA1" to "AOB007.SA1" compose a single track, TrackD.

#### **{17-5\_22-7} TKI\_BLK\_ATR**

The attributes of present TKI are written in the "TKI\_BLK\_ATR". In FIG. 22, the information shown within the broken lines extending from the TKI\_BLK\_ATR shows the bit composition of the TKI\_BLK\_ATR. In FIG. 22, the TKI\_BLK\_ATR is shown as being 16 bits long, with the bits from b3 to b15 being reserved for future use. The three bits from bit b2 to b0 are used to show the attributes of the TKI.

When one TKI corresponds to a complete track, the

value "00b" is written in the TKI\_BLK\_ATR (this setting is hereafter referred to as "Track"). When several TKIs correspond to the same track, the value "001b" is written in the TKI\_BLK\_ATR of the first TKI (this setting is hereafter referred to as "Head\_of\_Track"), the value "010b" is written in the TKI\_BLK\_ATRs of the TKIs that correspond to AOBs in the middle of the track (this setting is hereafter referred to as "Midpoint\_of\_Track"), and the value "011b" is written in the TKI\_BLK\_ATR of the TKI that corresponds to the AOB at the end of the track (this setting is hereafter referred to as "End\_of\_Track"). When a TKI is unused but a TKI region exists, which is to say, when there is a deleted TKI, the value "100b" is written in the TKI\_BLK\_ATR (this setting is hereafter referred to as "Unused"). When a TKI is unused and no TKI region exists, the value "101b" is written in the TKI\_BLK\_ATR.

#### **{17-5\_22-8\_19} Example Setting of the TKI\_BLK\_ATR**

The following describes the settings of the TKI\_BLK\_ATR for each TKI in the example shown in FIG. 19.

By referring to the TKI\_BLK\_ATR of each TKI, it can be seen that the four pairs TKI#1 ("AOB001.SA1"), TKI#2 ("AOB002.SA1"), TKI#3 ("AOB003.SA1"), and TKI#8 ("AOB008.SA1") each correspond to separate tracks since the TKI\_BLK\_ATR of each of TKI#1, TKI#2, TKI#3, and TKI#8 is set as "Track".





the AOB recorded in the AOB file corresponding to a TKI is written in the "TKI\_PB\_TM" in the TKI.

When a track is composed of a plurality of TKIs, the entire playback period of the track is written in the TKI\_PB\_TM of the first TKI corresponding to the track, while the playback period of the corresponding AOB is written into the second and following TKIs for the track.

#### **{17-5\_22-10} TKI\_AOB\_ATR**

The encoding conditions used when producing an AOB, which is to say information such as (1) the sampling frequency at which the AOB recorded in the corresponding AOB file was sampled, (2) the transfer bitrate, and (3) the number of channels, is written in the "TKI\_AOB\_ATR" in a TKI. The bit composition of the TKI\_AOB\_ATR is shown within the broken lines that extend from the "TKI\_AOB\_ATR" in FIG. 22.

In FIG. 22, the TKI\_AOB\_ATR is composed of 32 bits, with the coding mode being written in the four-bit field from bit b16 to bit b19. When the AOB is encoded according to MPEG-2 AAC (with ADTS header), the value "0000b" is written into this field, while when the AOB is encoded according to MPEG-layer 3 (MP3), the value "0001b" is written. When the AOB is encoded according to Windows Media Audio (WMA), the value "0010b" is written in this field.

The bitrate used when encoding the AOB is written

in the eight-bit field between bit b15 and bit b8. When the AOB is encoded according to MPEG-2 AAC (with ADTS header), a value between "16" and "72" is written into this field, while when the AOB is encoded according to MPEG1-layer 3 (MP3), a value between "16" and "96" is written. When the AOB is encoded according to MPEG1-layer 3 (MP3) LSF, a value between "16" and "80" is written into this field, while when the AOB is encoded according to Windows Media Audio (WMA), a value between "8" and "16" is written.

The sampling frequency used when encoding the AOB is written in the four-bit field between bit b7 and bit b4. When the sampling frequency is 48kHz, the value "0000b" is written in this field. When the sampling frequency is 44.1kHz, the value is "0001b", when the sampling frequency is 32kHz, the value "0010b", when the sampling frequency is 24kHz, the value "0011b", when the sampling frequency is 22.05kHz, the value "0100b", and when the sampling frequency is 16kHz, the value "0101b".

The number of channels is written in the three-bit field from bit b3 to bit b1. When one channel (i.e., monoaural) is used, the value "000b" is written in this field, while when two channels (i.e., stereo) is used, the value "001b" is written in this field.

The twelve-bit field from bit b31 to bit 20 is reserved for future use, as is the bit b0.

**{17-5\_22-11} ISRC**

An ISRC (International Standard Recording Code) is written in the TKGI. In FIG. 22, the broken lines extending from the "ISRC" box show the content of the ISRC. As shown  
5 in the drawing, the ISRC is composed of ten bytes, with a Recording-item code (#12) being written into the four-bit field between bit b4 and bit b7. A Recording code/Recording-item code (#11) is written in the four-bit field between bit b8 and bit b11.

10 A Recording Code (ISRC#10, #9, #8) is written in the twelve-bit field between bit b12 and bit b23. A Year-of-Recording code (ISRC#6, #7) is written in the eight-bit field b24 and bit b31.

The First Owner Code (ISRC #3, #4, #5) is written  
15 in the six-bit field between bit b32 and bit b37, the six-bit field between bit b40 and bit b45, and the six-bit field between bit b48 and bit b53. The Country Code (ISRC #1, #2, #3) is written in the six-bit field between bit b56 and bit b61 and the six-bit field between bit b64 and bit  
20 b69. A one-bit Validity flag is written in a one-bit field composed of bit b79. A detailed description of ISRC can be found in ISO3901:1986 "Documentation-International Standard Recording Code (ISRC)".

**{17-5\_22-12\_23A-1} BIT**

The "Block Information Table (BIT)" is a table for

managing an AOB\_BLOCK, and has the detailed composition shown in FIGS. 23A and 23B.

As shown in FIG. 23A, a BIT is composed of a DATA\_OFFSET field that occupies a region from the 60th byte to the 63rd byte, an SZ\_DATA field that occupies a region from the 64th  
5 byte to the 67th byte, a TMSRTE\_Ns field that occupies a region from the 68th byte to the 71st byte, an FNs\_1st\_TMSRTE field that occupies a region from the 72nd byte to the 73rd byte, an FNs\_Last\_TMSRTE that occupies a region from the  
10 74th byte to the 75th byte, an FNs\_Middle\_TMSRTE field that occupies a region from the 76th byte to the 77th byte, and a TIME\_LENGTH field that occupies a region from the 78th byte to the 79th byte.

Each of these fields is described in detail below.

#### **{17-5\_22-12\_23A-2} DATA\_Offset**

The relative address of the start of an AOB\_BLOCK from the boundary between clusters is written in the "DATA\_OFFSET" as a value given in byte units. This  
20 expresses the size of an invalid area between an AOB and the AOB\_BLOCK. As one example, when a user records a radio broadcast on a flash memory card 31 as AOBs and wishes to delete an intro part of a track over which a DJ has spoken, the DATA\_OFFSET in the BIT can be set to have the track  
25 played back without the part including the DJ's voice.

**{17-5\_22-12\_23A-3} SZ\_DATA**

The data length of an AOB\_BLOCK expressed in byte units is written in "SZ\_DATA". By subtracting a value produced by adding the SZ\_DATA to the DATA\_Offset from the file size (an integer multiple of the cluster size), the size of the invalid area that follows the AOB\_BLOCK can be found.

**{17-5\_22-12\_23A-4} TMSRTE\_Ns**

The total number of TMSRT\_Entries included in an AOB\_BLOCK is written in "TMSRTE\_Ns".

**{17-5\_22-12\_23A-5} "FNs\_1st\_TMSRTE",  
"FNs\_Last\_TMSRTE", "FNs\_Middle\_TMSRTE"**

The number of AOB\_FRAMES included in the AOB\_ELEMENT positioned at the start of a present AOB\_BLOCK is written in "FNs\_1st\_TMSRTE".

The number of AOB\_FRAMES included in the AOB\_ELEMENT positioned at the end of the present AOB\_BLOCK is written in "FNs\_Last\_TMSRTE".

The number of AOB\_FRAMES included in each AOB\_ELEMENT apart from those at the start and the end of the present AOB\_BLOCK, which is to say AOB\_ELEMENTS in the middle of the AOB\_BLOCK, is written in "FNs\_Middle\_TMSRTE".

The playback period of an AOB\_ELEMENT is written in the format shown in FIG. 23C in the "TIME\_LENGTH" field

to an accuracy in the order of milliseconds. As shown in  
FIG. 23C, the "TIME\_LENGTH" field is 16-bits long. When  
the encoding method used in MPEG-ACC or MPEG-Layer3, the  
playback period of an AOB\_ELEMENT is two seconds, so that  
5 the value "2000" is written in the "TIME\_LENGTH" field.

#### {17-5\_22-13\_23B}

FIG. 23B shows the number of AOB\_FRAMES indicated  
by "FNs\_Middle\_TMRTE". In the same way as FIG. 14, FIG.  
10 23B shows the relationship between the sampling\_frequency  
and the number of AOB\_FRAMES included in an AOB\_ELEMENT  
in the middle of an AOB\_BLOCK.

The relationship between the sampling\_frequency and  
the number of frames included in the AOB\_ELEMENT shown in  
15 FIG. 23B is the same as that shown in FIG. 14, which is  
to say, the number of frames in an AOB\_ELEMENT depends on  
the sampling frequency used. The number of frames written  
in "FNs\_1st\_TMSRTE" and "FNs\_Last\_TMSRTE" will  
fundamentally be the same as the number written in  
20 "FNs\_Middle\_TMSRTE", though when an invalid area is present  
in the AOB\_ELEMENTS at the start and/or end of an AOB\_BLOCK,  
the values given in "FNs\_1st\_TMSRTE" and/or  
"FNs\_Last\_TMSRTE" will differ from the value in  
"FNs\_Middle\_TMSRTE".

#### **{17-5\_22-14\_24} Example of a Stored AOB\_ELEMENT**

FIG. 24 shows the clusters 007 to 00E that store the AOB composed of AOB\_ELEMENT#1 to AOB\_ELEMENT#4. The following describes the settings in the BIT when an AOB is stored as shown in FIG. 24. AOB\_ELEMENT#1 to AOB\_ELEMENT#4 that are stored in cluster 007 to cluster 00E are indicated in FIG. 24 by the triangular flags, with TMSRT\_entries being set in the TKI for each of AOB\_ELEMENT#1 to AOB\_ELEMENT#4.

In this example, the first part of AOB\_ELEMENT#1 at the start of the AOB is stored in cluster 007, while the last part of AOB\_ELEMENT#4 at the end of the AOB is stored in cluster 00E. The AOB\_ELEMENTS #1 to #4 occupy the region between md0 in cluster 007 to md4 in cluster 00E. As shown by arrow sd1 in FIG. 24, the SZ\_DATA in the BIT indicates that AOB\_ELEMENTS #1 to #4 occupy a region from the start of cluster 007 to the end of cluster 00E, and so does not indicate that there are the invalid areas ud0 and ud1 in clusters 007 and 00E that are not occupied by an AOB\_ELEMENT.

On the other hand, the AOB also includes the parts ud0 and ud1 that are present in clusters 007 and 00E but are not occupied by AOB\_ELEMENT#1 or AOB\_ELEMENT#4. The DATA\_Offset given in the BIT gives the length of the unoccupied region ud0, which is to say, a position value for the start of the AOB\_ELEMENT#1 relative to the start of cluster 007.

In FIG. 24, the AOB\_ELEMENT#1 occupies a region from md0 in cluster 007 to md1 in cluster 008.

This AOB\_ELEMENT#1 does not occupy all of cluster 008, with the remaining part of the cluster being occupied by AOB\_ELEMENT#2. AOB\_ELEMENT#4 occupies a region from md3 midway through cluster 00C to md4 midway through cluster 00E. In this way, AOB\_ELEMENTS may be stored across cluster boundaries, or in other words, AOB\_ELEMENTS can be recorded without regard for the boundaries between clusters. The "FNs\_1st\_TMSRTE" in the BIT shows the number of frames in AOB\_ELEMENT#1 that is located in clusters 007 and 008, while the "FNs\_Last\_TMSRTE" in the BIT shows the number of frames in AOB\_ELEMENT#4 that is located in clusters 00C to 00E.

In this way, AOB\_ELEMENTS can be freely positioned without regard for the boundaries between clusters. The BIT provides information showing the offset from a cluster boundary to an AOB\_ELEMENT and the number of frames in each AOB\_ELEMENT.

#### **{17-5\_22-14\_25} Use of the Number of Frames given in each AOB\_ELEMENT (part 1)**

The following describes how the number of frames in each AOB\_ELEMENT given in the BIT is used. This number of frames given in the BIT is used when forward or backward search is performed. As mentioned earlier, such operations play back 240 milliseconds of data after first skipping



data with a playback period of two seconds.

FIG. 25 shows how AOB\_FRAME#x+1, which should be played back next, is set when performing forward search starting from an AOB\_FRAME#x in an AOB\_ELEMENT#y in an AOB.

FIG. 25 shows the case when a user selects forward search during the playback of AOB\_FRAME#x included in AOB\_ELEMENT#y. In FIG. 25, "t" represents the intermittent playback period (here, 240 milliseconds), "f(t)" shows the number of frames that correspond to this intermittent playback period, "skip\_time" shows the length of the period that should be skipped between intermittent playback periods (here, two seconds), "f(skip\_time)" shows the number of frames that correspond to this skip time. Intermittent playback is achieved by repeating the three procedures (1), (2), and (3) described below.

(1) The playback apparatus refers to the TMSRT\_entry in the TKTMSRT and jumps to the start of the flag symbol (AOB\_ELEMENT).

(2) The playback apparatus performs playback for 240 milliseconds.

(3) The playback apparatus jumps to the start of the next flag symbol (AOB\_ELEMENT).

The AOB\_FRAME#x+1 that exists 2s+240ms from the

AOB\_FRAME#x included in the AOB\_ELEMENT#y will definitely be present in the AOB\_ELEMENT#y+1. When specifying the AOB\_FRAME#x+1 that is 2s+240ms from the AOB\_FRAME#x, the first address of the next AOB\_ELEMENT#y+1 can be immediately  
5 calculated by reading a TMSRT\_entry from the TKTMSRT, though a playback apparatus cannot know the number of AOB\_FRAMES from the start address of the AOB\_ELEMENT#y+1 to the AOB\_FRAME#x+1 from the TMSRT\_entry alone.

To calculate this number of AOB\_FRAMES, it is  
10 necessary to subtract the total number of frames included in the AOB\_ELEMENT#y from the total of (1) the number#x showing the position of the AOB\_FRAME#x relative to the start of the AOB\_ELEMENT#y, (2) f(t) and (3) f(skip\_time). To simplify the calculation of the relative frame position  
15 of AOB\_FRAME#x+1 in AOB\_ELEMENT#y+1, the "FNs\_1st\_TMSRTE", "FNs\_Middle\_TMSRTE", and "FNs\_Last\_TMSRTE" for each AOB\_ELEMENT are written in the BIT, as mentioned above.

#### **{17-5\_22-15\_26A} Use of the Number of Frames given in each**

##### **20 AOB\_ELEMENT (part 2)**

The number of frames written in the BIT is also used when the playback apparatus performs a time search function where playback starts at a point indicated using a time code. In FIG. 26A, shows how a playback apparatus can  
25 specify the AOB\_ELEMENT and AOB\_FRAME corresponding to the playback start time indicated by the user. When playback

is to commence from a time indicated by the user, the indicated time (in seconds) is set in the `Jmp_Entry` field, and playback should begin from an `AOB_ELEMENT#y` and an `AOB_FRAME` position `x` that satisfy Equation 2 given below.

5

#### Equation 2

$$\text{Jmp\_Entry(sec)} = (\text{FNs\_1st\_TMSRTE} + \text{FNs\_middle\_TMSRTE} * y + x) * 20\text{msec}$$

10 Since the "`FNs_1st_TMSRTE`" and "`FNs_Middle_TMSRTE`" are provided in the BIT, these can be substituted into Equation 2 to calculate the `AOB_ELEMENT#y` and `AOB_FRAME#x`. Having done this, a playback apparatus can refer to the `TKTMSRT` of the AOB, calculate the first address of the

15 `AOB_ELEMENT#y+2` (which is the  $(y+2)^{\text{th}}$  `AOB_ELEMENT` in this AOB), and start the search for `AOB_FRAME#x` from this first address. On finding the  $x^{\text{th}}$  `AOB_FRAME`, the playback apparatus starts the playback from this frame. In this way, the playback apparatus can start the playback of data

20 from the time indicated by `Jmp_Entry` (in seconds).

In this way, a playback apparatus does not have to search for the ADTS header parts of `AOB_FRAMES`, and only needs to perform the search in `AOB_ELEMENTS` that are given in the `TMSRT_entries` in the `TKTMSRT`. This means that the

25 playback apparatus can find a playback position corresponding to an indicated playback time at high speed.

In the same way, when the `Jmp_Entry` is set and the time search function is used on a track that is composed of a plurality of AOBs, the playback apparatus only needs to calculate an `AOB_ELEMENT#y` and `AOB_FRAME#x` that satisfy Equation 3 below.

### Equation 3

`Jmp_Entry` (in seconds) =

Playback period from `AOB#1` to `AOB#n` +

(`FNs_1st_TMSRTE`(`#n+1`)+`FNs_middle_TMSRTE`(`#n+1`)\*`y`+`x`)\*20m sec

The total playback period of the AOBs from `AOB#1` to `AOB#n` is as follows.

Total Playback Period from `AOB#1` to `AOB#n` =

[`"Fns_1st_TMSRTE"`(`#1`)+`"Fns_Middle_TMSRTE"`(`#1`)\* (Number of `TMSRT_entries`(`#1`)-2) + `"Fns_Last_TMSRTE"`(`#1`) + `"Fns_1st_TMSRTE"` (`#2`)+(`"Fns_Middle_TMSRTE"`(`#2`)\* Number of `TMSRT_entries` (`#2`)-2) + `"Fns_Last_TMSRTE"`(`#2`) + `"Fns_1st_TMSRTE"` (`#3`) + (`"Fns_Middle_TMSRTE"`(`#3`)\* Number of `TMSRT_entries` (`#3`) -2)+`"Fns_Last_TMSRTE"` (`#3`) . . . + `"Fns_1st_TMSRTE"`(`#n`)+ (`"Fns_Middle_TMSRTE"` (`#n`) \* Number of `TMSRT_entries` (`#n`)-2) + `"Fns_Last_TMSRTE"`(`#n`)]  
\*20msec

Having calculated an `AOB#n`, an `AOB_ELEMENT#y`, and `AOB_FRAME#x` that satisfy Equation 3, the playback apparatus

refers to the TKTMSRT corresponding to the AOB#n+1, searches for the  $x^{\text{th}}$  AOB\_FRAME from the address at which the  $(y+2)^{\text{th}}$  AOB\_ELEMENT (i.e., AOB\_ELEMENT#y+2) is positioned, and starts the playback from this  $x^{\text{th}}$  AOB\_FRAME.

5

#### **{17-5\_22-16\_27A,B} Deletion of an AOB File and a TKI**

This completes the explanation of all of the information included in the TKI. The following describes how the TKI is updated in the following four cases. In the first case (case1), a track is deleted. In the second case (case2) a track is deleted and a new track is recorded. In the third case (case3) two out of a plurality of tracks are selected and combined into a single track. Finally, in the fourth case (case4), one track is divided to produce two tracks.

The following describes case1 where a track is deleted.

FIGS. 27A and 27B show the partial deletion of a track. The example in FIGS. 27A and 27B corresponds to the TrackManager shown in FIG. 19, and assumes that the user has indicated the partial deletion of Track B. The AOB corresponding to TrackB is recorded in "AOB002.SA1", which is associated with TKI#2. This means that the deletion of "AOB002.SA1" is accompanied by the setting of "Unused" into the TKI\_BLK\_ATR of TKI#2. This state where "AOB002.SA1" has been deleted and "Unused" has been set

into the TKI\_BLK\_ATR of TKI#2 is shown in FIG. 27B. Since "AOB002.SA1" has been deleted, the region that was formerly occupied by "AOB002.SA1" is freed to become an unused region. As mentioned above, the other change is that "Unused" is set in the TKI\_BLK\_ATR of TKI#2.

**{17-5\_22-17\_28A,B} Assignment of TKIs when a new AOB is recorded**

The following describes case2 where a new track is recorded after the deletion of a track.

FIG. 28A shows the TrackManager after the deletion of tracks has been performed several times. As shown in FIG. 28A, if the tracks corresponding to TKI#2, TKI#4, TKI#7, and TKI#8 have been deleted, then "Unused" is set in the TKI\_BLK\_ATR of these TKI. While AOB files are deleted in the same way as conventional data files, the TrackManager is updated by merely setting "Unused" in the TKI\_BLK\_ATR of the corresponding TKI. These means that TKIs whose TKI\_BLK\_ATRs are set at "Unused" can appear at different places in the TrackManager.

FIG. 28B shows how a new TKI and AOB file are written when a TKI whose TKI\_BLK\_ATR is "Unused" is present in the TrackManager. Like in FIG. 28A, the TKI#2, TKI#4, TKI#5, TKI#7, and TKI#8 in FIG. 28B are set as "Unused".

In FIG. 28B, the new track to be written is composed of four AOBs. The unused TKIs used to record these AOBs

are determined according to the DPL\_TK\_SRPs or can be freely chosen. In the present example, the unused TKIs numbered TKI#2, TKI#4, TKI#7, and TKI#8 are used to record the TKIs for the new track.

5        Since these four AOBs compose one track,  
"Head\_of\_Track" is set in the TKI\_BLK\_ATR of TKI#2,  
"Middle\_of\_Track" is set in the TKI\_BLK\_ATR of TKI#4 and  
TKI#7, and "End\_of\_Track" is set in the TKI\_BLK\_ATR of TKI#8.  
The TKI\_LNK\_PTR in each of the four TKIs, TKI#2, TKI#4,  
10    TKI#7, and TKI#8, used to compose the new TrackD is set  
so as to show the TKI forming the next part of TrackD, so  
that as shown by the arrows TL2, TL4, and TL7, TKI#4 is  
set in the TKI\_LNK\_PTR of TKI#2, TKI#7 is set in the  
TKI\_LNK\_PTR of TKI#4, and TKI#8 is set in the TKI\_LNK\_PTR  
15    of TKI#7.

After this, the files "AOB002.SA1", "AOB004.SA1",  
"AOB007.SA1", and "AOB008.SA1" having the same numbers as  
TKI#2, TKI#4, TKI#7, TKI#8 are produced, and the four AOBs  
composing TrackD are stored in these four files.

20        By appropriately setting the TKI\_LNK\_PTRs and  
TKI\_BLK\_ATRs, this fourth track TrackD can be managed using  
TKI#2, TKI#4, TKI#7, and TKI#8.

As described above, when a new track is written onto  
the flash memory card 31, TKIs in the TrackManager that  
25    are set as "Unused" are assigned as the TKIs to be used  
for tracks that are to be newly recorded.

**{17-5\_22-18\_29A,B} Setting of TKI when Combining Two Tracks**

The following describes the updating of the TKI when combining tracks (case3).

FIGS. 29A and 29B show how the TKIs are set when two tracks are combined to produce a new track. The example in FIG. 29A uses the same TrackManager as FIG. 19 and shows the case when the user performs an editing operation to combine TrackC and TrackE into a single track.

In this case, the AOBs that correspond to TrackC and TrackE are recorded in the AOB files "AOB003.SA1" and "AOB008.SA1" which correspond to TKI#3 and TKI#8, so that the TKI\_BLK\_ATRs of TKI#3 and TKI#8 are rewritten. FIG. 29B shows the TKI\_BLK\_ATR of these TKIs after rewriting.

In FIG. 29A, the TKI\_BLK\_ATRs of TKI#3 and TKI#8 is written as "Track", but in FIG. 29B the TKI\_BLK\_ATR of TKI#3 is rewritten to "Head\_of\_Track" and the TKI\_BLK\_ATR of TKI#8 is rewritten as "End\_of\_Track". By rewriting the TKI\_BLK\_ATRs in this way, the AOB files "AOB003.SA1" and "AOB008.SA1" which correspond to TKI#3 and TKI#8 end up being treated as parts of a single track, the new TrackC. This operation is accompanied by the TKI\_LNK\_PTR of TKI#3 being rewritten to indicate TKI#8.

It should be particularly noted here that while the TKI\_BLK\_ATRs in the TKI are rewritten, no processing is performed to physically combine the AOB files "AOB003.SA1"



and "AOB008.SA1". This is because AOB files are each encrypted using different FileKeys, so that when combining AOB files, it would be necessary to perform two processes for each AOB file to first decrypt the encrypted AOB file and then to re-encrypt the result, resulting in an excessive processing load. Also, an AOB file combined in this way would be encrypted using a single FileKey, which would make the combined track less secure than the tracks used to produce it.

The TKI is originally designed so as to suppress the size of the TKTMSRT, so that the physical combining of AOB files by an editing operation would also carry the risk of the TKI becoming too large.

For the reasons given above, editing operations that combine tracks leave the AOB files in their encrypted state and are achieved by merely changing the attributes given by the TKI\_BLK\_ATRs.

#### **{17-5\_22-18\_29A,B-1\_30, 31} Conditions That Should be Satisfied When Combining Tracks**

The combining of tracks is performed by changing the TKI\_BLK\_ATR attributes as described above, but the AOBs that are included in the combined tracks should satisfy the conditions given below.

A first condition is that the AOB that is to compose a latter part of a new track needs to have the same audio



accumulate in the buffer of the playback apparatus, though this cannot be achieved when there are consecutive Type2 AOBs. In such case, an underflow is likely to occur in the buffer of the playback apparatus, so that uninterrupted playback by the playback apparatus can no longer be guaranteed. Therefore, in order to avoid such underflows, the second condition stipulating that three or more Type2 AOBs cannot be linked continuously is used.

FIG. 30A shows a Type1 AOB, while FIG. 30B shows two examples of Type2 AOBs. In FIG. 30B, both AOBs are composed of less than two AOB\_ELEMENTS, with none of the AOB\_ELEMENTS including a number of AOB\_FRAMES that is set for an "FNs\_Middle\_TMSRTE". Since the absence of an AOB\_ELEMENT with the number of AOB\_FRAMES set for an "FNs\_Middle\_TMSRTE" is the condition by which an AOB is classified as a Type2 AOB, this means that all of the AOBs shown in this drawing are classified as Type2 AOBs.

In FIG. 31A, a combining of Type1+Type2+Type2+Type1 AOBs into a single track is shown. As this combining does not involve the linking of three Type2 AOBs, these AOBs may be linked to form a single track.

FIG. 31B shows the linking of Type1+Type2+Type2+Type2+Type1 AOBs into a single track. This combining would result in there being three consecutive Type2 AOBs, and so is prohibited.

**{17-5\_22-18\_29A,B-1\_32} Combining of Tracks with respect  
to combinations of Type1 and Type2 AOBs**

In the combining of AOBs into a single track shown  
in FIG. 31A, if the last AOB in the first track is a Type1  
5 AOB, the combining can be performed regardless of whether  
the first part of this track is a Type1 AOB or a Type2 AOB.  
FIG. 32A shows the case where the last AOB in the first  
track is a Type1 AOB and the first AOB in the next track  
is also a Type1 AOB. FIG. 32B shows the case where the  
10 last AOB in the first track is a Type1 AOB and the first  
AOB in the next track is a Type2 AOB. As the second condition  
is satisfied in both of these cases, the illustrated tracks  
can be combined into a single track.

When the last AOB in the first track is a Type2 AOB  
15 and the preceding AOB in the first track is a Type 1 AOB,  
this first track can be combined with a following track  
that starts with a Type1 AOB regardless of whether the first  
AOB in the first track is a Type1 AOB or a Type2 AOB.

FIG. 32C shows the case where the first track ends  
20 with a Type1 AOB and a Type2 AOB in that order and the second  
track starts with a Type1 AOB. FIG. 32D shows the case  
where the first track ends with a Type1 AOB and a Type2  
AOB in that order and the second track starts with a Type2  
AOB and a Type1 AOB in that order. As the second condition  
25 is satisfied in both of these cases, the illustrated tracks  
can be combined into a single track.



to the newly generated AOB file "AOB002.SA1".

#### **{17-5\_22-19\_33A,B-1\_34A,B} Updating of the Directory Entries and the FAT Values**

5        When the AOB file "AOB003.SA1" is divided to produce "AOB002.SA1" the directory entries and FAT values have to be updated. This updating is explained below. FIG. 34A shows how the SD-Audio Directory Entry in the SD-Audio Directory to which the AOB file "AOB003.SA1" belongs is  
10        written before the file is divided.

      The AOB file "AOB003.SA1" is divided into a plurality of parts that are stored in clusters 007, 008, 009, 00A . . . 00D, 00E. In this case, the first cluster number for the AOB file "AOB003.SA1" given in the directory entry is  
15        written as "007". The values (008), (009), (00A) . . . (00D), (00E) are also written in the FAT values 007, 008, 009, 00A . . . 00D corresponding to the clusters 007, 008, 009, 00A . . . 00D.

      When the AOB file "AOB003.SA1" is divided so that  
20        its latter part becomes the new AOB file "AOB002.SA1", a "filename", a "filename extension" and a "number of first clusters in file" for the new AOB file "AOB002.SA1" are added to the SD-Audio directory entry. FIG. 34B shows how the SD-Audio Directory Entry in the SD-Audio Directory to  
25        which the AOB file "AOB003.SA1" belongs is written after the AOB file "AOB003.SA1" has been divided.



TKI to produce a template for the new TKI, and then dividing/updating the TKTMSRT and BIT in this template and updating the remaining information fields.

FIG. 35A shows the case where an AOB\_FRAME in an AOB is divided. The first level in FIG. 35A shows the four AOB\_ELEMENTS, AOB\_ELEMENT#1, AOB\_ELEMENT#2, AOB\_ELEMENT#3, and AOB\_ELEMENT#4. The data lengths of these AOB\_ELEMENTS are set in the TKTMSRT as the four TMSRT\_entries #1, #2, #3, and #4. If the boundary bd1 for the division is set in AOB\_ELEMENT#2 in FIG. 35A, AOB\_ELEMENT#2 is divided into a first region (1) made up of the frames located before the boundary bd1 and a second region (2) composed of the frames located after the boundary bd1. FIG. 35B shows the two AOBs AOB#1 and AOB#2 obtained by dividing the AOB midway through AOB\_ELEMENT#2.

#### **{17-5\_22-19\_33A,B-3\_36} Setting of the BIT**

FIG. 36 shows how the BIT is set when an AOB is divided as shown in FIG. 35. The AOB shown in FIG. 35 is divided at the boundary bd1. The AOB#1 produced by this division includes the two AOB\_ELEMENTS AOB\_ELEMENT#1 and AOB\_ELEMENT#2, while the other AOB#2 produced by this division includes the three AOB\_ELEMENTS, AOB\_ELEMENT#1, AOB\_ELEMENT#2, and AOB\_ELEMENT#3.

In FIG. 36, these AOB\_ELEMENTS have also been given the triangular flags to show the settings of the





for AOB#2 is given as the amount of data from the start of cluster 00B to a point midway through cluster 00E plus the data length of the part of cluster 00F occupied by AOB\_ELEMENT#1.

5           The part of AOB\_ELEMENT#2 in AOB#1 that is included in the copy of cluster 00A stored in cluster 00F needs to be excluded from AOB#2, so that the DATA\_Offset field in the BIT of AOB#2 is set at the size of the part of AOB\_ELEMENT#2 in AOB#1 included in cluster 00F.

10           As can be seen from FIG. 36, the division of the AOB result in only the AOB\_ELEMENT that includes the boundary for the division being divided into two and in the other AOB\_ELEMENTS positioned before and after the divided AOB\_ELEMENT remaining unchanged. As a result, the  
15   "FN\_Last\_TMSRTE" of AOB#2 is set at the same value for the "AOB\_ELEMENT#4" before the division, and the "FNs\_1st\_TMSRTE" of AOB#2 is set at AOB\_ELEMENT#1 of AOB#2, which is to say, the number of frames included in the part that follows the boundary once AOB\_ELEMENT#2 has been  
20   divided.

#### **{17-5\_22-19\_33A,B-4\_37} Setting of the BIT**

FIG. 37 shows a more specific example of changes in the BITs as a result of the division of a track. The left  
25   side of FIG. 37 shows an example of the settings of the BIT before division. In this BIT, the Data\_Offset is set

as "X", the SZ\_DATA is set at "52428", and the TMSRTE\_Ns is set at "n". The FNs\_1st\_TMSRTE is set at "80 frames", the FNs\_Middle\_TMSRTE is set at "94 frames", and the FNs\_Last\_TMSRTE is set at "50 frames".

5       The right side of FIG. 37 shows the settings of two  
BITS produced by the division of a track. When the AOB  
corresponding to the BIT on the left side of FIG. 37 is  
divided as shown in FIG. 35A, the Data\_Offset in the BIT  
of the first track produced by the division is set at "X"  
10 like the track before division, the "SZ\_DATA" is updated  
to the data length "Q" from the start to the division point  
Q, and the TMSRTE\_Ns is set at "k" which shows the number  
of TMSRT\_entries from the first TMSRT\_entry to the k<sup>th</sup>  
TMSRT\_entry. The FNs\_1st\_TMSRTE and FNs\_Middle\_TMSRTE  
15 are respectively set at "80" and "94" frames in the same  
way as the BIT before division, but since the final  
AOB\_ELEMENT in the AOB of the first track produced by the  
division includes "p" AOB\_FRAMES, the FNs\_Last\_TMSRTE is  
set at "p frames."

20       In the BIT of the second track produced by the division,  
the "Data\_Offset" is set at "R", the "SZ\_DATA" is set at  
(original SZ#DATA "52428"-data length up to division point  
Q), and the TMSRTE\_Ns is set at "n-k+1" produced by adding  
one (for the kth TMSRT\_entry that is newly added as a result  
25 of the division) to the number of TMSRT\_entries from the  
k<sup>th</sup> TMSRT\_entry to the n<sup>th</sup> TMSRT\_entry.

The FNs\_Middle\_TMSRTE and FNs\_Last\_TMSRTE are set at the same values as the BIT before division, which is to say, "94 frames" and "50 frames" respectively.

The first AOB\_ELEMENT in the AOB of this second track includes "94-p" AOB\_FRAMES, so that "94-p" is set in the FNs\_1st\_TMSRTE of the BIT corresponding to this track.

#### **{17-5\_22-19\_33A,B-5\_38} Setting of the BIT**

FIG. 38 shows the TKTMSRT after division. The following explains the settings of the TMSRT first. The TMSRT of the first track includes the TMSRT\_entries from the first TMSRT\_entry of the AOB before division to the kth TMSRT\_entry, which is to say, the TMSRT\_entries #1 to #k.

It should be noted here that the AOB\_ELEMENT#k that includes the boundary for the division only includes region (1), so that the k<sup>th</sup> TMSRT\_entry only includes a data size corresponding to this region (1). The TMSRT of the second track includes the TMSRT\_entries from the kth TMSRT\_entry of the AOB before division to the n<sup>th</sup> TMSRT\_entry, which is to say, the TMSRT\_entries #k to #n. It should be noted here that the AOB\_ELEMENT#k that includes the boundary for the division only includes region (2), so that the k<sup>th</sup> TMSRT\_entry only includes a data size corresponding to this region (2).

The copying of the TKI is accompanied by the division

and updating of the TKTMSRT and the BIT, and once the remaining information has been updated, the TKIs for the new tracks produced by the division will be complete. In the same way as when combining tracks, the AOB files are not decrypted, so that two tracks can be produced by dividing an AOB file in its encrypted state. Since the division of an AOB file does not involve decryption and re-encryption, the processing load of dividing a track can be suppressed. This means that tracks can be edited even by a playback apparatus with limited processing power.

This completes the explanation of the TKI. The following describes the Playlists.

#### **{17-6} PlaylistManager**

As shown by the broken lines h5 in FIG. 17, the PlaylistManager shown is made up of PlaylistManager\_Information (PLMGI) for managing the Playlists stored in the flash memory card 31, Default\_Playlist\_Information (DPLI) for managing all of the track stored in the flash memory card 31, and PlaylistInformation (PLI) #1, #2, #3, #4 . . . #m. Each PLI is information for a user-defined Playlist. As shown by the broken lines h6, the DPLI is composed of Default\_Playlist\_General\_Information (DPLGI) and Default\_Playlist\_Track\_Search\_Pointers (DPL\_TK\_SRP) #1, #2, #3, #4 . . . #m. As shown by the broken lines h7, each

PLI is composed of Playlist\_General\_Information (PLGI), and Playlist\_Track\_Search\_Pointers (PL\_TK\_SRP) #1, #2, #3, #4 . . . #m.

The DPLI referred to here differs from each PLI in the following way. While the DPLI has to indicate all of the tracks stored in the flash memory card 52, a PLI does not have this restriction and can indicate any number of the tracks. This opens up various possibilities for the user. As representative examples, the user can generate Playlist\_Information indicating only his (her) favorite tracks and store this Playlist\_Information in the flash memory card 31, or can have a playback apparatus automatically generate Playlist\_Information that only indicates tracks of a certain genre, out of a plurality of tracks stored in the flash memory card 31, and store the resulting Playlist Information in the flash memory card 31.

#### **{17-7\_18} Number of Playlists and Their Data Sizes**

As shown in FIG. 18, a maximum of 99 Playlists can be stored on one flash memory card 31. The combined data size of the PlaylistManager\_Information (PLMGI) and the Default Playlist Information (DPLI) is also fixed at 2,560 bytes. Each PLI has a fixed length of 512 bytes. The "DPL\_TK\_SRP" included in the Default Playlist Information includes a "DPL\_TK\_ATR" and a "DPL\_TKIN". On the other

hand, the "PL\_TK\_SRP" field included in a PLI includes only a "PL\_TK\_SRP". The format of the DPL\_TK\_ATR, DPL\_TKIN, and PL\_TKIN fields is shown in FIG. 39.

5    **{17-8\_39-1} Format of DPL\_TK\_SRP**

FIG. 39A shows the format of the DPL\_TK\_SRP. In FIG. 39A, the DPL\_TKIN is written in the 0th to 9th bits in the DPL\_TK\_SRP, while the DPL\_TK\_ATR is written in the 13th to 15th bits. The 10th to 12th bits in the DPL\_TK\_SRP are reserved for future use.

The TKI number is written in the DPL\_TKIN that occupies the 0th to 9th bits in the DPL\_TK\_SRP. This enables a TKI to be specified.

15    **{17-9\_39B} Format of the PL\_TK\_SRP**

FIG. 39B shows the format of the PL\_TK\_SRP. This is a ten-bit field in which PL\_TKIN, which is to say, a TKI number, is written.

20    **{17-8\_39A-2} Composition of DPL\_TK\_ATR**

The broken lines h51 and h52 that extend from the DPL\_TK\_ATR in FIG. 39A show an example setting of the DPL\_TK\_ATR. As can be seen from this drawing, the DPL\_TK\_ATR is set for a DPL\_TK\_SRP in the same way as TKI\_BLK\_ATR is set for a TKI, which is to say, the DPL\_TK\_ATR is set at one of "Track", "Head\_of\_Track"

"Midpoint\_of\_Track", and "End\_of\_Track".

In more detail, when the TKI indicated by the TKIN is used and an Audio Object (AOB) corresponding to one complete track is recorded in the AOB file corresponding  
5 to the indicated TKI (i.e., when the TKI\_BLK\_ATR of the TKI is "Track"), the value "00b" is set in the "DPL\_TK\_ATR".

When the TKI indicated by the TKIN is used and an Audio Object (AOB) corresponding to only the start of a track is recorded in the AOB file corresponding to the  
10 indicated TKI (i.e., when the TKI\_BLK\_ATR of the TKI is "Head\_of\_Track"), the value "001b" is set in the "DPL\_TK\_ATR". When the TKI indicated by the TKIN is used and an Audio Object (AOB) corresponding to a midway part track is recorded in the AOB file corresponding to the  
15 indicated TKI (i.e., when the TKI\_BLK\_ATR of the TKI is "Midpoint\_of\_Track"), the value "010b" is set in the "DPL\_TK\_ATR". When the TKI indicated by the TKIN is used and an Audio Object (AOB) corresponding to an end part of a track is recorded in the AOB file corresponding to the  
20 indicated TKI (i.e., when the TKI\_BLK\_ATR of the TKI is "End\_of\_Track"), the value "011b" is set in the "DPL\_TK\_ATR".

Conversely, when the TKI indicated by the TKIN is unused and the TKI region is merely established, which  
25 corresponds to when a TKI has been deleted (i.e., when the TKI\_BLK\_ATR of the TKI is "Unused"), the value "100b" is



set in the DPL\_TK\_ATR.

When the TKI indicated by the TKIN is unused and no TKI region has been established, which is to say, when a TKI is in an initial state, the value "101b" is set in the  
5 "DPL\_TK\_ATR".

Since the number of a TKI is written in the DPL\_TKIN, it is clear which of the plurality of TKI corresponds to each DPL\_TK\_SRP. The position of the DPL\_TK\_SRP in the Default\_Playlist\_Information shows when the AOB  
10 corresponding to the TKI that in turn corresponds to the DPL\_TK\_SRP will be played back, i.e., the ordinal position of the AOB in the Default\_Playlist. As a result, the order of the DPL\_TK\_SRP items in the Default\_Playlist denotes the order in which a plurality of tracks will be played,  
15 or in other words, determines the playback order of tracks.

#### **{17-9\_40-1} Interrelationship Between the**

##### **Default\_Playlist\_Information, TKI, and AOB files**

FIG. 40 shows the interrelationship between the  
20 Default\_Playlist\_Information, the TKI, and the AOB files. The second, third, and fourth levels in this drawing are the same as the first, second, and third levels in FIG. 19, and so show a TrackManager including eight TKI and eight AOB files. FIG. 40 differs from FIG. 19 in that a box showing  
25 the Default\_Playlist\_Information is given on the first level. The eight small divisions shown in this box show

the eight DPL\_TK\_SRP included in the Default\_Playlist\_Information. The upper part of each division shows the DPL\_TK\_ATR, while the lower part shows the DPL\_TKIN.

5 As shown by the arrows DT1, DT2, DT3, DT4 . . . in FIG. 40, DPL\_TK\_SRP#1 and TKI#1 are related, as are DPL\_TK\_SRP#2 and TKI#2, DPL\_TK\_SRP#3 and TKI#3, and DPL\_TK\_SRP#4 and TKI#4.

Looking at the DPL\_TK\_ATR fields in the DPL\_TK\_SRP,  
10 it can be seen that "Track" has been set for each of DPL\_TK\_SRP#1, DPL\_TK\_SRP#2, DPL\_TK\_SRP#3, and DPL\_TK\_SRP#8. In other words, the four combinations DPL\_TK\_SRP#1 → TKI#1("AOB001.SA1"), DPL\_TK\_SRP#2 → TKI#2("AOB002.SA1"), DPL\_TK\_SRP#3 → TKI#3("AOB003.SA1"),  
15 DPL\_TK\_SRP#8 → TKI#8("AOB008.SA1") correspond to four separate tracks.

Meanwhile, none of DPL\_TK\_SRP#4, DPL\_TK\_SRP#5, DPL\_TK\_SRP#6, and DPL\_TK\_SRP#7 has a DPL\_TK\_ATR set as "Track". Instead, the DPL\_TK\_SRP#4 of DPL\_TK\_ATR is set  
20 at "Head\_of\_Track", the DPL\_TK\_ATR of DPL\_TK\_SRP#7 is set at "End\_of\_Track" and the DPL\_TK\_ATRs of DPL\_TK\_SRP#5 and DPL\_TK\_SRP#6 are set at "Midpoint\_of\_Track".

This means that TKI#4("AOB004.SA1"), which is related to DPL\_TK\_SRP#4, is the start of a track,  
25 TKI#5("AOB005.SA1") and TKI#6("AOB006.SA1"), which are respectively related to DPL\_TK\_SRP#5 and DPL\_TK\_SRP#6, are

middle parts of a track, and TKI#7("AOB007.SA1"), which is related to DPL\_TK\_SRP#7, is the end of a track.

The DPL\_TK\_SRP entries in the DefaultPlaylist show in what order the AOBs corresponding to each TKI are to be played back. The DPL\_TKINS of DPL\_TK\_SRP#1, #2, #3, #4 . . . #8 in the DefaultPlaylist of FIG. 40 indicate TKI#1, #2, #3, #4 . . . #8. As shown by the arrows (1) (2) (3) (4) . . . (8), the AOB file "AOB001.SA1" corresponding to TKI#1 will be played back first, "AOB002.SA1" corresponding to TKI#2 will be played back second, "AOB003.SA1" corresponding to TKI#3 will be played back third, and "AOB004.SA1" corresponding to TKI#4 will be played back fourth.

#### **{17-10\_41} Example Settings for the DefaultPlaylist and Playlist\_Information**

FIG. 41 shows example settings for the Default\_Playlist and the Playlist\_Information using the same notation as FIG. 40. In FIG. 41, the box on the first level shows the Default\_Playlist, while the three boxes on the second level show the PLIs.

The small divisions in the box showing the Default\_Playlist shows the eight DPL\_TK\_SRP values included in the Default\_Playlist, while the small divisions in the boxes illustrating each PLI show three or four PL\_TK\_SRP values. The setting of the TKIN of each DPL\_TK\_SRP included in the Default\_Playlist\_Information

is the same as in FIG. 40. However, the settings of the TKIN of the PL\_TK\_SRP included in each PLI are completely different to those in the DPL\_TK\_SRP.

5    **{17-10\_42} Correspondence between the DPL\_TK\_SRP and the TKI**

FIG. 42 shows the correspondence between the DPL\_TK\_SRP and the TKI using the same notation as in FIG. 40. In FIG. 42, Playlist#1 is composed of PL\_TK\_SRP#1, #2, #3. Of these, #3 is written as the PL\_TKIN of PL\_TK\_SRP#1, while #1 is written as the PL\_TKIN of PL\_TK\_SRP#2 and #2 as the PL\_TKIN of PL\_TK\_SRP#3. This means that when tracks are played back according to Playlist#1, a plurality of AOBs will be played back as shown by the arrows (11) (12) (13) in the order AOB#3, AOB#1, AOB#2.

Playlist#2 is composed of PL\_TK\_SRP#1, #2, #3. Of these, #8 is written as the PL\_TKIN of PL\_TK\_SRP#1, while #3 is written as the PL\_TKIN of PL\_TK\_SRP#2 and #1 as the PL\_TKIN of PL\_TK\_SRP#3. This means that when tracks are played back according to Playlist#2, a plurality of AOBs will be played back, as shown by the arrows (21) (22) (23) in the order AOB#8, AOB#3, AOB#1, which is to say, in a completely different order to Playlist#1.

Playlist#3 is composed of PL\_TK\_SRP#1, #2, #3, #4. the PL\_TKIN of these PL\_TK\_SRP#1 to #4 are respectively set as #8, #4, #3, and #1. This means that when tracks

are played back according to Playlist#3, a plurality of AOBs will be played back as follows. First, AOB#8 that composes TrackE is played back as shown by the arrow (31). Next, AOB#4, AOB#5, AOB#6, and AOB#7 that compose TrackD are played back as shown by the arrow (32). After this, AOB#3 and AOB#1 that respectively compose TrackC and TrackA are played back as shown by the arrows (33) and (34).

Of special note here is that when a track is composed of a plurality of TKI, only the TKI number of the start of the track is written into the PL\_TK\_SRP entry. In more detail, while the DPL\_TK\_SRP values given in the Default\_Playlist\_Information specifies the four TKIs (TKI#4,TKI#5,TKI#6,TKI#7) that compose TrackD, the PL\_TK\_SRP given in a set of Playlist\_Information does not need to indicate all four TKIs. For this reason, PL\_TK\_SRP#2 in Playlist#3 only indicates TKI#4 out of TKI#4 to TKI#7.

On the other hand, a DPLI including a plurality of DK\_TK\_SRP has a data size that is no greater than one sector and is always loaded into the RAM of a playback apparatus. When tracks are played back according to a Playlist, the playback apparatus refers to the DK\_TK\_SRPs that are loaded into its RAM and so can search for TKIs at high speed. To playback TKIs (AOBs) using a PL\_TK\_SRP that only indicates the TKI number of the first TKI, a playback apparatus searches the DPL TK SRP loaded in its RAM based on the TKI

indicated by the PL\_TK\_SRP and judges whether the current track is composed of a plurality of TKI. If so, the playback apparatus executes the appropriate procedure for playing back all of the corresponding TKIs (AOBs).

5 As described above, the Default\_Playlist and a plurality of PLIs are written in the Playlist\_Manager. If different playback orders are written in the DPL\_TKINs and PL\_TKINs of the DPL\_TK\_SRP and PL\_TK\_SRP composing such playlists, it becomes possible to play back AOBs in  
10 different orders. By offering a variety of playback orders to the user in this way, the user can be given the impression of there being a number of music albums stored in the flash memory card 31.

Of special note here is that the data size of the  
15 DPL\_TK\_SRP corresponding to an AOB file is small (at no more than two bytes), while the data size of the TKI corresponding to an AOB file is large (at up to 1,024 bytes). When reordering the TKI in the TrackManager, a large number of accesses need to be made to the flash memory card 31,  
20 but when the DPL\_TK\_SRP are reordered in the Default\_Playlist\_Information or a PLI, this can be performed with fewer accesses to the flash memory card 31.

In view of this, when the navigation data is edited, the order of the DPL\_TK\_SRP in the Default\_Playlist is  
25 actively changed in accordance with the editing operation, while the order of the TKI in the TrackManager is left

unchanged in spite of the editing operation.

#### **{17-9\_40-2\_43A,B} Reordering of the DPL\_TK\_SRP**

The following describes an editing operation that  
5 changes the playback order of tracks by reordering the  
DPL\_TK\_SRPs in the Default\_Playlist\_Information. FIGS.  
43A and 43B show one example of the reordering of tracks.  
The settings of the DPL\_TK\_SRPs and TKIs in FIG. 43A are  
the same as in FIG. 40.

10 In FIG. 40A, the DPL\_TKIN in DPL\_TK\_SRP#3 is set at  
TKI#3, while the DPL\_TKIN in DPL\_TK\_SRP#8 is set at TKI#8.  
The following describes the case when these DPL\_TK\_SRPs  
with the thick outlines in FIG. 40A are interchanged.

The numbers (1) (2) (3) (4) (5) (6) (7) (8) in FIG. 43B show  
15 the playback order of tracks after this editing operation.  
It should be noted here that while the playback order shown  
in FIG. 43A is TrackA, TrackB, TrackC, TrackD, TrackE, in  
FIG. 43B the DPL\_TKINS of DPL\_TK\_SRP#3 and DPL\_TK\_SRP#8  
are interchanged in the Default\_Playlist\_Information, so  
20 that the tracks will be played back in the order TrackA,  
TrackB, TrackE, TrackD, TrackC. In this way, the playback  
order of tracks can be easily changed by changing the order  
of the DPL\_TK\_SRPs in the Default\_Playlist\_Information.

While the above explanation deals with an editing  
25 operation that changes the order of tracks, the following  
will describe the following four operations that were

explained with respect to the changes in the TKIs. These operations are a first case (case1) where a track is deleted, a second case (case2) where a new track is recorded, a third case (case3) where two freely selected tracks are combined to produce a new track, and a fourth case (case4) where a track is divided to produce two new tracks.

#### **{17-9\_40-3\_44A,B} Deletion of a Track**

The following describes case1 where a track is deleted.

FIGS. 44A and 44B show how the Default\_Playlist, TrackManager, and AOB files are updated when, out of the DefaultPlaylist shown in FIG. 40, DPL\_TK\_SRP#2 and TKI#2 are deleted. In these drawings, the same part of an AOB is deleted as in FIG. 27 that was used to describe the deletion of a TKI. As a result, the second, third, and fourth levels in FIG. 44A and 44B are the same as in FIG. 27. The difference with FIG. 27 is that Default\_Playlist\_Information including a plurality of DPL\_TK\_SRP#s is given on the first level, in the same way as FIG. 40.

The present example deals with the case when the user deletes TrackB composed of DPL\_TK\_SRP#2→TKI#2 ("AOB002.SA1") that is shown with the thick outline in FIG. 44A. In this case, DPL\_TK\_SRP#2 is deleted from Default\_Playlist\_Information and DPL\_TK\_SRP#3 to DPL\_TK\_SRP#8 are each moved up by one place in the playback



order so as to fill the place in the order freed by the deletion of DPL\_TK\_SRP#2.

When the DPL\_TK\_SRP#s are moved up in this way, the final DPL\_TK\_SRP#8 is set as "Unused". On the other hand,  
5 the TKI corresponding to the deleted part is set as "Unused" as shown in FIGS. 27A and 27B without other TKIs being moved to fill the gap created by the deletion. Deletion of the TKI is also accompanied by the deletion of the AOB file "AOB002.SA1".

10 In this way, DPL\_TK\_SRP#s are moved up in the playback order but TKIs are not moved, so that in FIG. 44B only the DPL\_TKINs in the DPL\_TK\_SRP#s are updated. For this example, the DPL\_TKIN in DPL\_TK\_SRP#2 is set so as to indicate TKI#3 as shown by the arrow DT11, the DPL\_TKIN in DPL\_TK\_SRP#3  
15 is set so as to indicate TKI#4 as shown by the arrow DT12, the DPL\_TKIN in DPL\_TK\_SRP#4 is set so as to indicate TKI#5, and the DPL\_TKIN in DPL\_TK\_SRP#5 is set so as to indicate TKI#6. The DPL\_TKIN in DPL\_TK\_SRP#8 that has been set at "Unused" is set so as to indicate TKI#2, as shown by the  
20 arrow DT13.

When a track is deleted, the DPL\_TK\_SRP used for following tracks in the playback order are moved up, while the TKI corresponding to the deleted track is set at "Unused" while remaining in its present position. In this way, an  
25 editing operation is not accompanied by movement of TKIs, which suppresses the processing load when editing tracks.

**{17-9\_40-4\_45A,B} Assignment of TKIs when Recording Tracks**

The following describes case2 when a new track is recorded following the partial deletion of a track. FIGS. 45A and 45B show how an operation that writes a new TKI and DPL\_TK\_SRP is performed when an "Unused" TKI and DPL\_TK\_SRP are present.

These drawings are largely the same as FIGS. 28A and 28B that were used to explain the assignment of a new TKI to a TKI set at "Unused". The second, third, and fourth levels in FIGS. 45A and 45B are the same as the first three levels in FIGS. 28A and 28B. The difference between these drawings is that the first levels in FIGS. 45A and 45B show the Default\_Playlist\_Information composed of a plurality of DPL\_TK\_SRP. In FIG. 45A, the DPL\_TK\_SRP#4 to DPL\_TK\_SRP#8 are set as "Unused". On the other hand, in FIG. 28 the TKI#2, TKI#4, TKI#5, TKI#7, TKI#8 are set as "Unused".

While TKIs set at "Unused" are present here and there in the TrackManager, the "Unused" DPL\_TK\_SRP are positioned next to one another in the Default\_Playlist\_Information. This results from the used DPL\_TK\_SRP being moved up in the Default\_Playlist\_Information as described above, while no such moving up is performed for TKIs.

The following explanation describes the case when

TrackD composed of four AOBs is written. The TKIs for these four AOBs are respectively written into the following "Unused" TKIs in the TrackManager: TKI#2; TKI#4; TKI#7; and TKI#8.

5       The DPL\_TK\_SRP#s for these four AOBs are written in DPL\_TK\_SRP#4 to DPL\_TK\_SRP#7 in the Default\_Playlist\_Information. Since these four AOBs compose a single track, the DPL\_TK\_ATR of DPL\_TK\_SRP#4 is set at "Head\_of\_Track", the DPL\_TK\_ATRs of DPL\_TK\_SRP#5 and DPL\_TK\_SRP#6 are set at "Middle\_of\_Track", and the DPL\_TK\_ATR of DPL\_TK\_SRP#7 is set at "End\_of\_Track".

10       The DPL\_TKIN of DPL\_TK\_SRP#4 is set at TKI#2, the DPL\_TKIN of DPL\_TK\_SRP#5 at TKI#4, the DPL\_TKIN of DPL\_TK\_SRP#6 at TKI#7, and the DPL\_TKIN of DPL\_TK\_SRP#7 at TKI#8.

15       By setting the DPL\_TKINs and DPL\_TK\_ATRs in this way, TKI#2, TKI#4, TKI#7 and TKI#8 are managed as the fourth track TrackD.

20       In the above processing, a write is performed for "Unused" TKIs, though this has no effect on the other TKIs TKI#1, TKI#2, TKI#3, and TKI#4, as was also the case in FIGS. 28A and 28B.

### **{17-9\_40-5\_46A,B} Case3: Combining Tracks**

25       The following describes the updating of the Default\_Playlist\_Information when tracks are combined

(i.e., in case3). FIGS. 46A and 46B show one example of the combining of tracks.

These drawings are largely the same as FIGS. 29A and 29B that were used to explain the combining of TKIs. The second, third, and fourth levels in FIGS. 46A and 46B are the same as the first two levels in FIGS. 29A and 29B. The difference between these figures is that the first levels in FIGS. 46A and 46B show Default\_Playlist\_Information, in which DPL\_TK\_SRP#8 is set at "Unused" and is related to TKI#2 that is also set at "Unused". When an editing operation combining tracks is performed for AOB files and TKIs as shown in FIGS. 29A and 29B, the contents of DPL\_TK\_SRP#3 to DPL\_TK\_SRP#6 are each moved down by one and the content of DPL\_TK\_SRP#7 that is shown with the thick outline is copied into DPL\_TK\_SRP#3 as shown in FIGS. 46A and 46B. The TKIs are also updated, as shown in FIGS. 29A and 29B.

#### **{17-9\_40-6\_47A,B} Case4: Division of a Track**

The following describes the updating of the Default\_Playlist\_Information when a track is divided (case4).

FIGS. 47A and 47B show one example of the division of a track. These drawings are largely the same as FIGS. 33A and 33B that were used to explain the division of TKIs. The second and third levels in FIGS. 47A and 47B are the

same as the first two levels in FIGS. 33A and 33B. The difference between these figures is that the first level in FIGS. 47A and 47B shows Default\_Playlist\_Information, in which DPL\_TK\_SRP#8 is set at "Unused" and is related  
5 to TKI#2 that is also set at "Unused".

If, as in FIGS. 33A and 33B, the user indicates the division of TKI#3 ("AOB003.SA1") shown with the thick outline into two, the positions of DPL\_TK\_SRP#3 to DPL\_TK\_SRP#7 are each moved down by one in the order, and  
10 a DPL\_TK\_SRP set at "Unused" is moved within the Default\_Playlist\_Information to the former position of DPL\_TK\_SRP#3.

This new DPL\_TK\_SRP#3 is associated with the TKI, TKI#2, newly produced by the division. The AOB file  
15 "AOB002.SA1" associated with TKI#2 stores what was originally the latter part of the AOB file "AOB003.SA1". DPL\_TK\_SRP#2 is present before the DPL\_TK\_SRP#3 that is associated with TKI#2 and is associated with TKI#2 and "AOB002.SA1".

20 This is to say, "AOB002.SA1" and "AOB003.SA1" respectively store the latter and former parts of the original "AOB003.SA1", with the DPL\_TK\_SRP#2 and DPL\_TK\_SRP#3 corresponding to these files indicating that these AOBs are to be played back in the order "AOB003.SA1"  
25 and "AOB002.SA1". As a result, the latter and former parts of the original "AOB003.SA1" will be played back in the

order former part, latter part in accordance with the  
playback order given in the DPL\_TK\_SRP.

#### **{17-9\_40-8} Application of the Editing Processing**

5 By combining the above four editing processes, a user  
can perform a great variety of editing operations. When,  
for example, a recorded track has an intro over which a  
disc jockey has talked, the user can first divide the track  
to separate the part including the disc jockey's voice.  
10 The user can then delete this track to leave the part of  
the track that does not include the disc jockey.

This completes the explanation of the navigation data.  
The following describes a playback apparatus with a suitable  
composition for playing back the navigation data and  
15 presentation data described above.

#### **{48-1} External Appearance of the Playback Apparatus**

FIG. 48 shows a portable playback apparatus for the  
flash memory card 31 of the present invention. The playback  
20 apparatus shown in FIG. 48 has an insertion slot for  
inserting the flash memory card 31, a key panel for receiving  
user indications for operations such as playback, forward  
search, backward search, fast forward, rewind, stop etc.,  
and an LCD (liquid crystal display) panel. In terms of  
25 appearance, this playback apparatus resembles other kinds  
of portable music players.

The key panel includes:

a "Playlist" key that receives the selection of a playlist or a track;

5 a "|<<" key that receives a skip operation that moves the playback position to a start of the current track;

a ">>|" key that receives a skip operation that moves the playback position to a start of the next track;

10 a "<<" key and a ">>" key that respectively receive a backward search operation and a forward search operation enable the user to have the playback move quickly through the current track;

a "Display" key that receives an operation to have still images stored on the flash memory card 31 displayed;

15 a "Rec" key that receives a recording operation; an "Audio" key for receiving user selections of the sampling frequency or of stereo or monoaural is to be used;

20 a "Mark" key that receives user indications that mark positions in tracks; and

an "Edit" key that receives user indications for the editing of tracks or for the input of track titles.

#### {48-2} Improvements Made in This Portable Playback

#### 25 Apparatus for the Flash Memory Card 31

The differences between this portable playback

apparatus of the flash memory card 31 and a conventional portable music player lie in the following four improvements (1) to (4).

(1) A list of playlist and tracks is shown on the LCD panel to allow the user to indicate the Default\_Playlist\_Information, a PLI, or separate tracks.

(2) Keys on the key panel are assigned to the playlists and/or tracks displayed on the LCD panel to allow the user to select a track or playlist that is to be played back or edited.

(3) A time code showing a position in a track is displayed on the LCD panel 5 when a track is played back.

(4) A jog dial is provided to enable the user to set a time code for use as playback start time when using the time search function or as a division boundary when dividing a track.

#### **{48-2\_49\_50} Improvement (2)**

The following describes improvement (2) in detail.

FIG. 49 shows one example of a display screen shown on the LCD panel when the user selects a playlist, while FIGs. 50A to 50E show examples of the displayed content when the user selects a track.

In FIG. 49, the ASCII character strings "DEFAULTPLAYLIST", "PLAYLIST#1", "PLAYLIST#2", "PLAYLIST#3", and "PLAYLIST#4" represent the default



playlist and the four playlists stored in the flash memory card 31.

Meanwhile, the ASCII character strings "Track#1", "Track#2", "Track#3", "Track#4", "Track#5" represent the five tracks that are indicated in the playback order given by the default playlist stored in the flash memory card 31. In FIGS. 49 and 50A, the highlighted Playlist and track show the track or Playlist that is currently indicated for playback or editing.

If the user presses the ">>" key when Track#1 is indicated for playback within a playback order given by the default Playlist displayed on the LCD panel, Track#2 will be indicated for playback within the list of tracks, as shown in FIG. 50B. If the user presses the ">>" key again, Track#3 will be indicated for playback within the list of tracks, as shown in FIG. 50C.

If the user presses the "<<" key when Track#3 is indicated for playback within a playback order given by the default Playlist displayed on the LCD panel, Track#2 will be indicated for playback within the list of tracks, as shown in FIG. 50D. As shown in FIG. 50E, if the user presses the "Play" key when any of the tracks is indicated, the playback of the indicated track will begin, while if the user presses the "Edit" key, the indicated track will be selected for editing.

#### {48-3\_51} Improvement (4)

The following describes improvement (4) in detail. FIGS. 51A to 51C show an example operation of the jog dial. When the user rotates the jog dial by a certain amount, the playback time code displayed on the LCD panel will be increased or decreased in accordance with this certain amount. The example in FIG. 51A shows the case where the playback time code that is initially displayed on the LCD panel is "00:00:20".

When the user rotates the jog dial counterclockwise as shown in FIG. 51B, the playback time code is reduced to "0:00:10" in keeping with the amount by which the jog dial was rotated. Conversely, when the user rotates the jog dial clockwise as shown in FIG. 51C, the playback time code is increased to "0:00:30" in keeping with the amount by which the jog dial was rotated.

By allowing the user to change the playback time code in this way, the playback apparatus enables the user to indicate any playback time code in a track by merely rotating the jog dial. If the user then presses the "Play" key, AOBs will be played back starting from a position found according to Equation 2 and Equation 3.

By using the jog dial during a track dividing operation, the user can make fine adjustments to the playback time code used as the division boundary.

### **{52-1} Internal Construction of the Playback Apparatus**

The following describes the internal construction of the playback apparatus. This internal construction is shown in FIG. 52.

5 As shown in FIG. 52, the playback apparatus includes a card connector 1 for connecting the playback apparatus to the flash memory card 31, a user interface unit 2 that is connected to the key panel and the jog dial, a RAM 3, a ROM 4, a LCD panel 5 having a list frame for displaying  
10 a list of tracks or playlists and a playback time code frame for displaying a playback time code, an LCD driver 6 for driving the first LCD panel 5, a descrambler 7 for decrypting AOB\_FRAMEs using a different FileKey for each AOB file, an AAC decoder 8 for referring to the ADTS of an AOB\_FRAME  
15 descrambled by the descrambler 7 and decoding the AOB\_FRAME to obtain PCM data, a D/A converter 9 for D/A converting the PCM data and outputting the resulting analog signals to a speaker or headphone jack, and a CPU 10 for performing overall control over the playback apparatus.

20 As can be understood from this hardware construction, the present playback apparatus has no special hardware elements for processing the TrackManager and Default\_Playlist\_Information. To process the TrackManager and Default\_Playlist\_Information, a DPLI  
25 holding area 11, a PLI storing area 12, a TKI storing area 13, a FileKey storing area 14, and a double buffer 15 are

provided in the RAM 3, while a playback control program and an editing control program are stored in the ROM 4.

#### **{52-2} DPLI Holding Area 11**

5       The DPLI holding area 11 is an area for continuously holding Default\_Playlist\_Information that has been read from a flash memory card 31 connected to the card connector 1.

#### **{52\_12} PLI Storing Area 12**

10       The PLI storing area 12 is an area that is reserved for storing Playlist\_Information that has been selected for playback by the user.

#### **{52-3} TKI Storing Area 13**

15       The TKI storing area 13 is an area that is reserved for storing only the TKI corresponding to the AOB file that is currently indicated for playback, out of the plurality of TKI included in the TrackManager. For this reason, the  
20       capacity of the TKI storing area 13 is equal to the data size of one TKI.

#### **{52-4} FileKey Storing Area 14**

25       The FileKey storing area 14 is an area that is reserved for storing only the FileKey corresponding to the AOB file that is currently indicated for playback, out of the

plurality of FileKeys included in "AOBSA1.KEY" in the authentication region.

#### **{52-5} Double Buffer 15**

5       The double buffer 15 is an input/output buffer that is used when an input process, which successively inputs cluster data (data that is stored in one cluster) read from the flash memory card 31, and an output process, which reads AOB\_FRAMES from cluster data and successively outputs the AOB\_FRAMES to the descrambler 7, are performed in parallel.

10       The double buffer 15 successively frees the regions that were occupied by cluster data that has been outputted as AOB\_FRAMES and so secures regions for storing the next clusters to be read. This is to say, regions in the double buffer 15 are cyclically secured for storing cluster data using ring pointers.

#### **{52-5\_53\_54A,B} Input and Output by the Double Buffer 15**

20       FIG. 53 shows how input and output are performed for the double buffer 15. FIGS. 54A and 54B show how regions in the double buffer 15 are cyclically secured for storing cluster data using a ring pointers.

25       The arrows pointing downward and to the left are pointers to write addresses for cluster data, which is to say, the write pointer. The arrows pointing upward and to the left are pointers to read addresses for cluster data,

which is to say, the read pointer. These pointers are used as the ring pointer.

#### **{54-6\_53}**

5        When a flash memory card 31 is connected to the card connector 1, cluster data in the user region of the flash memory card 31 is read out and stored in the double buffer 15 as shown by the arrows w1 and w2.

10        The read cluster data is successively stored into the positions in the double buffer 15 shown by the write pointers wp1 and wp2.

#### **{52-7\_54A}**

15        Of the AOB\_Frames included in the cluster data stored in this way, the AOB\_Frames present at the positions ① ②③④⑤⑥⑦⑧⑨ that are successively indicated by the read pointer are outputted one at a time to the descrambler 7 as shown by the arrows r1, r2, r3, r4, r5 ....

20        In the present case, the cluster data 002 and 003 is stored in the double buffer 15 and the read positions ① ②③④ are successively indicated by the read pointer, as shown in FIG. 53. When the read pointer reaches the read position ⑤, all of the AOB\_FRAMES included in cluster 002 will have been read, so that cluster 004 is read and, as  
25        shown by the arrow w6 in FIG. 54A, is overwritten into the region that was previously occupied by cluster 002.

**{52-8\_54B}**

The read pointer then advances to the read positions ⑥ and ⑦, and eventually reaches the read position ⑨, at which point all of the AOB\_FRAMEs included in cluster 003 will have been read, so that cluster 005 is read and, as shown by the arrow w7 in FIG. 54B, is overwritten into the region that was previously occupied by cluster 003.

The output of an AOB\_FRAME and the overwriting of cluster data are repeatedly performed as described above, so that the AOB\_FRAMEs included in an AOB file are all successively outputted to the descrambler 7 and AAC decoder 8.

**{52-9\_55-58} Playback Control Program Stored in the ROM**

**4**

The following describes the playback control program stored in the ROM 4.

FIG. 55 is a flowchart showing the processing in the AOB file reading procedure. FIGS. 56, 57, and 58 are flowcharts showing the processing in the AOB\_FRAME output procedure.

**{52-9\_55-1}**

These flowcharts use the variables w, z, y, and x. The variable w indicates one of the plurality of DPL\_TL\_SRPs. The variable z indicates an AOB file recorded in the user

region, the TKI corresponding to this AOB file, and the AOB included in this AOB file. The variable y indicates an AOB\_ELEMENT included in the AOB#z indicated by the variable z. The variable x indicates an AOB\_FRAME included  
5 in the AOB\_ELEMENT#y indicated by the variable y. The following first explains the processing in the AOB file read procedure, with reference to FIG. 55.

**{52-9\_55-2}**

10 In step S1, the CPU 10 reads the PlaylistManager and displays a list including the Default\_Playlist\_Information and the PLIs.

In step S2, the CPU 10 waits for an indication to play back AOBs in accordance with either the

15 Default\_Playlist\_Information or one of the PLIs.

When the Default\_Playlist\_Information is indicated, the processing moves from step S2 to step S3 where the variable w is initialized (#w←1) and then to step S4 where the TKI#z indicated by the DPL\_TKIN corresponding to  
20 DPL\_TK\_SRP#w in the Default\_Playlist\_Information is specified and only this TKI#z is read from the flash memory card 31 and stored into the TKI storing area 13.

In step S5, an AOB file#z with the same number as TKI#z is specified. In this way, the AOB file that is to be played  
25 back is finally specified.

The specified AOB file is in an encrypted state and



needs to be decrypted, so that steps S6 and S7 are performed.

In step S6, the playback apparatus accesses the authentication region and reads the FileKey#z that is stored in a FileKey\_Entry#z in the encryption key storing file, the FileKey\_Entry#z having the same number as the specified AOB file. In step S7, the CPU 10 sets the FileKey#z in the descrambler 7. This operation results in the FileKey being set in the descrambler 7, so that by successively inputting AOB\_FRAMES included in the AOB file into the descrambler 7, the AOB\_FRAMES can be successively played back.

#### **{52-9\_55-3}**

After this, the playback apparatus successively reads the clusters that store the AOB file. In step S8, the "first cluster number in the file" is specified for the AOB\_file#z in the directory entry. In step S9, the CPU 10 reads the data stored in this cluster from the flash memory card 31. In step S10, the CPU 10 judges whether the cluster number in the FAT value is "FFF". If not, in step S11 the CPU reads the data stored in the cluster indicated by the FAT value, before returning to step S10.

When the playback apparatus reads the data stored in any of the clusters and refers to the FAT value corresponding to this cluster, the processing in steps S10 and S11 will be repeated so long as the FAT value is not

set at "FFF". This results in the playback apparatus successively reading clusters indicated by the FAT values. When the cluster number given by a FAT value is "FFF", this means that all of the clusters composing the AOB file#z have been read, so that the processing advances from step S10 to step S12.

**{ 52-9\_ 55-4 }**

In step S12, the CPU 10 judges whether the variable#w matches the total number of DPL\_TK\_SRP's. If not, the processing advances to step S13, where the variable#w is incremented (#w-#w+1) before the processing returns to step S4. In step S4, the playback apparatus specifies TKI#z which is indicated by the DPL\_TKIN#w of DPL\_TK\_SRP#w in the Default\_Playlist\_Information, and writes only TKI#z into the TKI storing area 13. The TKI that was used up to this point will be still stored in the TKI storing area 13, though this current TKI will be overwritten by TKI#z that is newly read by the CPU 10.

This overwriting results in only the latest TKI being stored in the TKI storing area 13. Once the TKI has been overwritten, the processing in steps S5 to S12 is repeated for the AOB file#z. Once this processing has read all of the TKI and AOB files corresponding to all of the DPL\_TK\_SRPs included in the Default\_Playlist\_Information, the variable #z will match the total number of DPL\_TK\_SRP so that the judgement "Yes" is given in step S12 and the processing

in this flowchart ends.

#### **{52-9\_56\_57\_58} Output Processing for an AOB\_FRAME**

In parallel with the AOB file reading procedure, the  
5 CPU 10 performs the AOB\_FRAME output procedure in accordance  
with the flowcharts shown in FIGS. 56, 57, and 58. In these  
flowcharts, the variable "play\_time" shows how long  
playback has been performed for a current track, which is  
to say, the playback time code. The time displayed in the  
10 playback time code frame on the LCD panel 5 is updated in  
accordance with changes to this playback time code.  
Meanwhile, the variable "play\_data" represents the length  
of the data has been played back for the current track.

#### **15 {52-9\_56-1}**

In step S21, the CPU 10 monitors whether cluster data  
for the AOB file #z has accumulated in the double buffer  
15. This step S21 will be repeatedly performed until  
cluster data has accumulated, at which point the processing  
20 advances to step S22 where the variables x and y are  
initialized (#x←1, #y←1). After this, in step S23 the CPU  
10 searches the clusters for AOB file #z and detects the  
AOB\_FRAME#x in the AOB\_ELEMENT#y that is positioned no  
earlier than the Data\_Offset given in the BIT#z included  
25 in TKI#z. In this example, it is assumed that the seven  
bytes starting from the SZ\_DATA are occupied by the ADTS

header. By referring to the ADTS header, the data length indicated by the ADTS header can be recognized as audio data. The audio data and ADTS header are read together and are outputted to the descrambler 7. The descrambler  
5 7 decrypts the AOB\_FRAMEs, which are then decoded by the AAC decoder 8 and reproduced as audio.

#### **{52-9\_56-2}**

After this detection, in step S24 the AOB\_FRAME#x is  
10 outputted to the descrambler 7, and in step S25 the variable play\_time is incremented by the playback period of the AOB\_FRAME#x and the variable play\_data is incremented the amount of data corresponding the AOB\_FRAME#x. Since the playback time of AOB\_FRAME is 20msec in the present case,  
15 20msec is added to the variable "play\_time".

Once the first AOB\_FRAME has been outputted to the descrambler 7, in step S26 the playback apparatus refers to the ADTS header of AOB\_FRAME#x and specifies where the next AOB\_FRAME is. In step S27, the playback apparatus  
20 increments the variable#x ( $\#x \leftarrow \#x + 1$ ) and sets AOB\_FRAME#x as the next AOB\_FRAME. In step S28, AOB\_FRAME#x is inputted into the descrambler 7. After this, in step S29, the variable play\_time is incremented by the playback period of the AOB\_FRAME#x and the variable play\_data is incremented  
25 the amount of data corresponding the AOB\_FRAME#x. After incrementing AOB\_FRAME#x, in step S30 the CPU 10 judges

whether the variable #x has reached the value given in  
FNS\_1st\_TMSRTE.

If the variable #x has not reached the value in  
FNS\_1st\_TMSRTE, in step S31 the playback apparatus checks  
5 whether the user has pressed any key aside from the "Play"  
key, and then returns to step S26. The playback apparatus  
hereafter repeats the processing in steps S26 to S31 until  
the variable #x reaches the value in FNS\_1st\_TMSRTE or until  
the user presses any key aside from the "Play" key.

10 When the user presses a key aside from the "Play"  
key, the processing in this flowchart ends and suitable  
processing for the pressed key is performed. When the  
pressed key is the "Stop" key, the playback procedure stops,  
while when the pressed key is the "Pause" key, the playback  
15 is paused.

#### **{52-9\_57-1}**

On the other hand, when the variable #x reaches the  
value in FNS\_1st\_TMSRTE, the judgement "Yes" is made in  
20 step S30, and the processing proceeds to step S32 in FIG.  
57. Since all of the AOB\_FRAMES included in the present  
AOB\_ELEMENT will have been inputted into the descrambler  
7 in the processing between step S26 to S30, in step S32  
the variable #y is incremented to set the next AOB\_ELEMENT  
25 as the data to be processed and the variable #x is initialized  
(#y←#y+1, #x←1).

After this, in step S33 the playback apparatus refers to the TKTMSRT and calculates the first address of the AOB\_ELEMENT#y.

The playback apparatus then performs the procedure made up of steps S34 to S42. This procedure reads the AOB\_FRAMES included in an AOB\_ELEMENT one after another, and so can be said to resemble the procedure made up of steps S24 to S31. The difference with the procedure made up of steps S24 to S31 is the condition by which the procedure made up of steps S24 to S31 ends is whether the variable #x has reached the value shown by "FNs\_1st\_TMSRTE", while the condition by which procedure made up of steps S34 to S42 ends is whether the variable #x has reached the value shown by "FNs\_Middle\_TMSRTE".

When the variable #x reaches the value shown by "FNs\_Middle\_TMSRTE", the loop procedure made up of steps S34 to S42 ends, the judgement "Yes" is given in step S41 and the processing advances to step S43. In step S43, the CPU 10 increments the variable #y and initializes the variable #x ( $\#y \leftarrow \#y + 1$ ,  $\#x \leftarrow 1$ ). After this, in step S44 the variable y judges whether the variable #y has reached a value that is equal to one less than the TotalTMSRT\_entry\_Number in the TMSRT\_Header in the TKI#z.

When the variable #y is lower than (TotalTMSRT\_entry\_Number-1), the AOB\_ELEMENT#y is not the final AOB\_ELEMENT, so that the processing returns from step

009250" T090250  
S44 to step S32 and the loop procedure of step S32 to step  
S42 is performed. When the variable #y reaches  
(TotalTMSRT\_entry\_Number-1) the read procedure can be  
assumed to have proceeded as far as the penultimate  
5 AOB\_ELEMENT, so that the judgement "Yes" is given in step  
S44 and the processing advances to step S45 in FIG. 58.

#### {52-9\_57-2}

10 The procedure composed of steps S45 to S54 resembles  
the procedure composed of steps S33 to S42 in that each  
of the AOB\_FRAMES in the final AOB\_ELEMENT are read.

15 The difference with the procedure composed of steps  
S33 to S42 is that while the loop procedure composed of  
steps S33 to S42 ends when it is judged in step S41 that  
the variable #x has reached the value in "FNs\_Middle\_TMSRTE",  
the loop procedure composed of steps S45 to S54 ends when  
it is judged in step S53 that the variable #x has reached  
the value in "FNs\_Last\_TMSRTE" and the variable play\_data  
showing the size of the data that has hitherto been read  
20 has reached the value given as "SZ\_DATA".

The procedure composed of steps S49 to S54 is repeated  
until the conditions in step S53 are satisfied, at which  
point the judgement "Yes" is given in step S53 and the  
processing advances to step S55. In step S55, the CPU 10  
25 increments the variable #z ( $\#z \leftarrow \#z + 1$ ) before the processing  
returns to step S21 where the CPU 10 waits for the next





it to "00:00:00.040", as shown in FIG. 59C. In the same way, when the playback of AOB\_FRAME#6 ends, the playback period 20msec of AOB\_FRAME#6 is added to the playback time code to update it to "00:00:00.120", as shown in FIG. 59D.

5 This completes the description of the AOB\_FRAME output procedure.

In step S31 of the flowchart in FIG. 56, if the user presses a key aside from the "Play" key, the processing in this flowchart is terminated. The processing that  
10 accompanies a pressing of "Stop" or "Pause" key has already been described, though when the user presses one of the keys provided to have the playback apparatus perform special playback, the processing in this flowchart, or in the flowcharts shown in FIGS. 56, 57, or 58 is terminated and  
15 suitable processing for the pressed key is performed.

The following describes the procedure executed by the CPU 10 (1) when performing the forward search function in response to the user pressing the ">>" key and (2) when performing the time search function in response to the user  
20 operating the jog dial after pressing the "Pause" or "Stop" key.

#### **{52-10\_60} Forward Search Function**

FIG. 60 is a flowchart showing the procedure executed  
25 by the CPU 10 when performing the forward search function. When the user presses the ">>" key, the judgement "Yes"

is given in step S31, step S42 or step S54 in the flowcharts in FIGS. 56, 57 and 58 and the CPU 10 performs the processing in the flowchart of FIG. 60.

In step S61, the AOB\_FRAMES #x to #(x+f(t)-1) are  
5 inputted into the descrambler 7. Here "t" represents the intermittent playback period, f(t) represents the number of frames corresponding to the intermittent playback period, and d(t) represents the amount of data corresponding to the intermittent playback period. In step S62, the  
10 variable play\_time showing the playback elapsed time, and the variable play\_data showing the playback data amount are respectively updated using intermittent playback period "t", the number of frames f(t) corresponding to intermittent playback period, and the amount of data d(t)  
15 corresponding to the intermittent playback period  
(x←x+f(t), play\_time←play\_time+t,  
play\_data←play\_data+d(t)). Note that the intermittent playback period will generally be 240 msec (equivalent to the playback period of twelve AOB\_FRAMES).

20  
**{52-10\_60-1\_61A,B}**

FIGS. 61A and 61B show the incrementing of the playback time code during a forward search operation. FIG. 61A shows the initial value of the playback time code, with the  
25 playback point being the AOB\_FRAME#1 in AOB\_ELEMENT#51.

The playback time code in this case is "00:00:01.000".

When the first to twelve AOB\_FRAMES have been inputted into the descrambler 7 as the intermittent playback period, the playback period of twelve AOB\_FRAMES (i.e., 240msec) is added to the playback time code so that the playback time  
5 code becomes "00:00:01.240", as shown in FIG. 61B.

**{52-10\_60-2}**

After this updating, in step S63 the CPU 10 compares the incremented variable #x with the total number of frames  
10 in AOB\_ELEMENT#y and judges whether the incremented variable #x is within the total number of frames in AOB\_ELEMENT#y.

As mentioned earlier, the number of frames in an AOB\_ELEMENT positioned at the start of an AOB is  
15 "FNs\_1st\_TMSRTE", the number of frames in an AOB\_ELEMENT positioned in a central part of an AOB is "FNs\_Middle\_TMSRTE", and the number of frames in an AOB\_ELEMENT positioned at the end of an AOB is "FNs\_Last\_TMSRTE".

20 The CPU 10 performs the above judgement by comparing an appropriate one of these values with the variable #x. When the variable x is not within the present AOB\_ELEMENT#y, the CPU 10 then judges in step S64 whether there is an AOB\_ELEMENT that follows the AOB\_ELEMENT#y.

25 When the AOB\_ELEMENT#y is the final AOB\_ELEMENT in an AOB\_BLOCK, there will be no AOB\_ELEMENT that follows



**{52-10\_60-4\_61C}**

As shown in FIG. 61C, the intermittent skip period is added to the variable#x showing a frame position within the AOB\_ELEMENT#51. When the updated variable #x exceeds the number of frames in AOB\_ELEMENT#51, the variable #y is updated to indicate the next AOB\_ELEMENT and the number of frames in the AOB\_ELEMENT#51 is subtracted from the variable #x. As a result, the variable#x will now indicate a frame position within the AOB\_ELEMENT#52 indicated by the updated variable #y. The value 2.000 (=2sec) is then added to the present value "00:00:01.240" of the playback time code so that it becomes "00:00:03.240". The variable #x is updated by calculating  $(3240\text{msec}-2000\text{msec})/20\text{msec}$  to give the value "62", and so indicates the AOB\_FRAME#62 in the AOB\_ELEMENT#52.

**{52-10\_60-5\_61(d)}**

Once the AOB\_FRAME#62 in the AOB\_ELEMENT#52 has been inputted into the descrambler 7, the playback time code is updated as shown in FIG. 61D by adding "0.240" to the present value of "00:00:03.240" to give "00:00:03.480".

In step S67, the variables are updated in accordance with the intermittent skip time and then the processing in steps S68 to S71 are performed. This processing in steps S68 to S71 is the same as the processing in steps S63 to S66 and so updates the variable#x by a number of frames



the Default\_Playlist\_Information are displayed and the user indicates a desired track. When this track has been indicated and the user has operated the jog dial, the playback time code is updated. If the user then presses  
5 the "Play" key, the playback time code at that point is used to set a value in the variable "Jmp\_Entry" in seconds.

A judgement is then made as to whether the indicated track is composed of a plurality of AOBs or a single AOB. When the track is composed of a single AOB, the variables  
10 #y and #x are calculated so as to satisfy Equation 2. After this, a search for the AOB\_FRAME#x is started from the address in the (y+2)<sup>th</sup> position in the TKTMSRT corresponding to this AOB. Once this AOB\_FRAME#x has been found, playback starts from AOB\_FRAME#x.

15

#### {52-12}

When the track is composed of a plurality of AOBs, the variables #n (indicating an AOB), #y and #x are calculated so as to satisfy Equation 3. After this, a search  
20 for the AOB\_FRAME#x is started from the address in the (y+2)<sup>th</sup> position in the TKTMSRT corresponding to AOB#n. Once this AOB\_FRAME#x has been found, playback starts from AOB\_FRAME#x.

The following describes the case when playback is  
25 commenced from an arbitrary position with an AOB where the "FNs\_1st\_TMSRTE" in the BIT is "80 frames",

"FNs\_Middle\_TMSRTE" in the BIT is "94 frames", and the  
"FNs\_Last\_TMSRTE" in the BIT is "50 frames".

**{52-13\_62A,B}**

5 As one specific example of when the time search function  
is used, the following describes how the AOB\_ELEMENT and  
frame position from which playback should start are  
specified when a playback time code is indicated using the  
jog dial.

10 As shown in FIG. 62A, the user holds the playback  
apparatus in his/her hand and rotates the jog dial with  
his/her right thumb to indicate the playback time code  
"00:04:40.000 (=280sec)". When the BIT in the TKI for this  
AOB is as shown in FIG. 62B, Equation 2 is used as follows

15

$$\begin{aligned} 280\text{sec} &= (\text{FNs\_1st\_TMSRTE} + (\text{FNs\_Middle\_TMSRTE} * y) + x) * 20\text{msec} \\ &= (80 + (94 * 148) + 8) * 20\text{msec} \end{aligned}$$

so that the Equation 2 is satisfied for the values  
20 y=148 and x=8.

Since y=148, the entry address of the AOB\_ELEMENT#150  
(=148+2) is obtained from the TKTMSRT. Playback from the  
indicated playback time code 00:04:40.000 (=280.00sec) can  
then be performed by starting the playback at the eighth  
25 AOB\_FRAME from this entry address.



### **{52-14\_63\_64\_65}**

This completes the explanation of the processing of the CPU 10 in response to the user pressing the "Play" key. The following describes the editing control program stored in the ROM 4. This editing control program is executed when the user presses the "Edit" key, and contains the procedures shown in FIGS. 63, 64, and 65. The following describes the processing in this program with the flowcharts shown in these drawings.

#### **{52-14\_63-1} Editing Control Program**

When the user presses the "Edit" key, an interactive screen is displayed in step S101 in FIG. 63 to ask the user which of the three fundamental editing operations "deletion", "division" and "combining" is to be performed. In step S102, the CPU 10 judges what operation has been made by the user in response to the interactive screen. In the present example, it is assumed that the "|<<" and ">>|" keys on the key panel are also used as indicating "Up" and "Down" cursor operations, (i.e., these keys are used as "Up" and "Down" cursor keys). When the user indicates a "deletion" operation, the processing proceeds to the loop procedure composed of steps S103 and S104.

In step S103, the CPU 10 judges whether the user has pressed the "|<<" or ">>|" key. In step S104, the CPU 10 judges whether the user has pressed the "Edit" key. When

the user has pressed the "|<<" or ">>|" key, the processing advances from step S103 to S105, where the indicated track is set as the track to be edited. On the other hand, when the user has pressed the "Edit" key, the indicated track is set as a track to be deleted. The processing shown in FIG. 44 is executed, so that the TKI\_BLK\_ATR of each TKI for the indicated track is set at "Unused" to delete the indicated track.

#### **{52-14\_63-2} Combining Process**

When the user selects the combining process, the processing proceeds from step S102 to the loop procedure composed of steps S107 to S109. In the loop procedure composed of steps S107 to S109, the playback apparatus receives user inputs via the "|<<", ">>|", and "Edit" keys. When the user presses the "|<<" or ">>|" key, the processing advances from step S107 to step S110 where the indicated track is highlighted on the display. When the user presses the "Edit" key, the judgement "Yes" is given in step S108 and the processing advances to step S111. In step S111, the currently indicated track is set as the first track to be used in this editing process and the processing returns to the loop procedure composed of steps S107 to S109.

When a second track has been selected for editing, the judgement "Yes" is given in step S109, and the processing advances to step S112. In step S112, the CPU 110 refers

to the BITs in the TKIs of the former and the latter tracks and judges what kind of AOBs (Type1 or Type2) are present at the respective start and end of each of these tracks and tracks on either side of these tracks, if present.

5       After identifying the type of each relevant AOB, in step S113 the CPU 10 judges whether the arrangement of AOBs matches a certain pattern. When the arrangement of AOBs matches one of the four patterns shown in FIG. 32A to 32D where it is clear that three Type2 AOBs will not be present  
10       consecutively after the combining, the former and latter tracks are combined into a single track in step S115.

          In the other words, the operation shown in FIG. 46 is performed for the TKI and DPL\_TK\_SRP corresponding to these AOBs. By rewriting the TKI\_BLK\_ATRs in the TKIs,  
15       the plurality of tracks selected for editing are combined into a single track. When the arrangement of AOBs does not match any of the patterns in FIGS. 32A to 32D, meansing that there will be three or more Type2 AOBs after the combining, the CPU 10 judges that the combined track may  
20       cause a buffer underflow and so terminates the combining process.

#### **{52-14\_64-1} Track Division Process**

          When the user indicates that a track is to be divided,  
25       the processing advances from step S102 to the loop procedure composed of steps S116 to S117. In the loop procedure

composed of steps S116 to S117, the playback apparatus receives user inputs via the "|<<", ">>|", and "Edit" keys.

When the user presses the "|<<" or ">>|" key, the processing advances from step S116 to step S118 where the indicated track is set as the track to be edited. When  
5 the user presses the "Edit" key, the judgement "Yes" is given in step S117 and the processing advances to step S119.

In step S119, the indicated track is determined as the track to be edited and the processing advances to step  
10 S120 where the playback of this track is commenced. In step S121, the playback apparatus receives a user input via the "Mark" key.

When the user presses the "Mark" key, the playback of the track is paused and the processing advances to the  
15 loop procedure composed of steps S122 and S123. In step S122, the playback apparatus receives user operations made via the jog dial. When the user rotates the jog dial, the playback time code is updated in step S124 in accordance with the rotation of the jog dial.

20 After this, the loop procedure composed of steps S122 and S123 is repeated. If the user presses the "Edit" key, the processing proceeds from step S123 to step S125, where the playback time code displayed when the user pressed the "Edit" key is set as the division boundary. Note that an  
25 "Undo" function may be provided for this setting of the division boundary to allow the user to invalidate the

selected division boundary.

After this, the processing explained with reference to FIG. 47 is executed in step S126 to update the DPLI and TKI so as to divide the selected track.

5

#### **{52-14\_65-1} Process Setting a Playlist**

When the user chooses to set a Playlist, the processing switches to the procedure shown by the flowchart in FIG. 65. In this flowchart, the variable k given in this flowchart is used to indicate the position of a track in the playback order given by the Playlist that is being edited. The flowchart in FIG. 65 starts with this variable k being initialized to "1" in step S131, before the processing advances to the loop procedure composed of steps S132 to S134.

15

In the loop procedure composed of steps S132 to S134, the playback apparatus receives user operations made via the "|<<", ">>|", "Edit", and "Stop" keys. When the user presses the "|<<" or ">>|" key, the processing advances from step S132 to step S135 where a new track is indicated in accordance with the pressing of the "|<<" or ">>|" key. If the user presses the "Edit" key, the judgement "Yes" is given in step S133 and the processing advances to step S136.

20

25

In step S136, the track indicated when the user presses the "Edit" key is selected as the kth track in the playback

order. After this, in step S137 the variable k is incremented and the processing returns to the loop procedure composed of steps S132 to S134. This procedure is repeated so that the second, third and fourth tracks are successively selected. If the user presses the "Stop" key have specified several tracks that are to be played back in the specified order as a new Playlist, the processing advances from step S134 to step S138 where a PLI composed of PL\_TK\_SRP's that specify the TKIs corresponding to these tracks is generated.

#### **{66-1} Recording Apparatus**

The following describes one example of a recording apparatus for the flash memory card 31. FIG. 66 shows one example of a recording apparatus. This recording apparatus can be connected to the Internet, and is a standard personal computer that can perform reception when an encrypted SD-Audio directory is sent via communication lines to the recording apparatus by an electronic music distribution service, or when an audio data transport stream is sent via communication lines to the recording apparatus by an electronic music distribution service.

#### **{67-1} Hardware Composition of the Recording Apparatus**

FIG. 67 shows the hardware composition of the present recording apparatus.

As shown in FIG. 67, the recording apparatus includes

a card connector 21 for connecting the recording apparatus to the flash memory card 31, a RAM 22, a non-removable disk apparatus 23 for storing a recording control program that performs overall control over the recording apparatus, an A/D converter 24 that A/D converts audio inputted via a microphone to produce PCM data, an ACC encoder 25 for encoding the PCM data in units of a fixed time and assigning ADTS headers to produce AOB\_FRAMES, a scrambling unit 26 for encrypting the AOB\_FRAMES using a different FileKey for each AOB\_BLOCK, a modem apparatus 27 for receiving an audio data transport stream when an encrypted SD-Audio directory is sent via communication lines to the recording apparatus by an electronic music distribution service, or when an audio data transport stream is sent via communication lines to the recording apparatus by an electronic music distribution service, a CPU 28 for performing overall control over the recording apparatus, a keyboard 29 for receiving inputs made by the user, and a display 30.

**{67-2} Input Circuits RT1 to RT4**

When an encrypted SD-Audio directory, which is to be written in the data region and the authentication region, is sent via communication lines to the recording apparatus by an electronic music distribution service, the recording apparatus can write the encrypted SD-Audio directory into

the data region and authentication region of the flash memory card 31 as soon as the encrypted SD-Audio directory has been properly received.

However, (1) when an audio data transport stream that is not in the form of SD-Audio directory is sent to the recording apparatus by an electronic music distribution service, (2) when data is inputted into the recording apparatus in PCM format, or (3) when analog audio is recorded by the recording apparatus, the recording apparatus uses the following four input routes to write an audio data transport stream onto the flash memory card 31.

As shown in FIG. 67, the four input routes RT1, RT2, RT3, and RT4 are used to input an audio data transport stream when an audio data transport stream is stored in the flash memory card 31.

### **{67-3} Input Route RT1**

The input route RT1 is used when an encrypted SD-Audio directory is sent via communication lines to the recording apparatus by an electronic music distribution service, or when an audio data transport stream is sent via communication lines to the recording apparatus by an electronic music distribution service. In this case, the AOB\_FRAMES included in the transport stream are encrypted so that a different FileKey is used for the AOB\_FRAMES in different AOBs. Since there is no need to encrypt or encode



an encrypted transport stream, the SD-Audio directory or audio data transport stream can be stored directly into the RAM 22 in its encrypted state.

#### 5    **{67-4} Input Route RT2**

Input route RT2 is used when audio is inputted via a microphone. In this case, the audio inputted via the microphone is subjected to A/D conversion by the A/D converter 24 to produce PCM Data. The PCM data is then encoded  
10 by the AAC encoder 25 and assigned ADTS headers to produce AOB\_FRAMES. After this, the scrambling unit 26 encrypts the AOB\_FRAMES using a different FileKey for each AOB\_FRAMES in different AOB\_FILES to produce encrypted audio data. After this, the encrypted audio data is stored in the RAM  
15 22.

#### **{67-5} Input Route RT3**

Input route RT3 is used when PCM data read from a CD is inputted into the recording apparatus. Since data is  
20 inputted in PCM format, the data can be inputted as it is into the AAC encoder 25. This PCM data is encoded by the ACC encoder 25 and assigned ADTS headers to produce AOB\_FRAMES.

After this, the scrambling unit 26 encrypts the  
25 AOB\_FRAMES using a different FileKey for the AOB\_FRAMES in different AOBs to produce encrypted audio data. After

this, the encrypted audio data is stored in the RAM 22.

#### **{67-6} Input Route RT4**

The input route RT4 is used when a transport stream  
5 inputted via one of the three input routes RT1, RT2, and  
RT3 is written into the flash memory card 31.

This storing of audio data is accompanied by the  
generation of TKIs and Default\_Playlist\_Information. In  
the same way as the playback apparatus, the main functioning  
10 of the recording apparatus is stored in the ROM. This is  
to say, a recording program that includes the characteristic  
processing of the recording apparatus, which is to say,  
the recording of AOBs, the TrackManager, and the  
PlaylistManager, is stored in the non-removable disk  
15 apparatus 23.

#### **{67-7\_68} Processing of the Recording Apparatus**

The following describes the processing in the  
recording procedure that writes a transport stream in the  
20 flash memory card 31 via the input routes RT1, RT2, RT3  
and RT4, with reference to the flowchart in FIG. 68 that  
shows this processing.

The variables "Frame\_Number" and "Data\_Size" used  
in this flowchart are as follows. The variable  
25 Frame\_Number is used to manage the total number of  
AOB\_FRAMES that have already been recorded in an AOB\_FILE.

The variable Data\_Size is used to manage the data size of the AOB\_FRAMES that have already been recorded in the AOB\_FILE.

The processing in this flowchart starts in step S200  
5 with the CPU 28 generating the DefaultPlaylist and the TrackManager. In step S201, the CPU 28 initializes the variable #z ( $z \leftarrow 1$ ). In step S202, the CPU 28 generates the AOB\_FILE#z and stores it in the data region of the flash memory card 31. At this point, the filename, filename  
10 extension, and first cluster number for the AOB\_FILE#z will be set in a directory entry in the SD\_Audio Directory in the data region. After this, in step S203, the CPU 28 generates TKI#z and stores it in the TrackManager. In step S204, the CPU 28 generates the DPL\_TK\_SRP#w and stores it  
15 in the Default\_Playlist\_Information. After this, in step S205 the CPU 28 initializes the variable#y ( $y \leftarrow 1$ ) and in step S206, the CPU 28 initializes the Frame\_Number and Data\_Size ( $\text{Frame\_Number} \leftarrow 0, \text{Data\_Size} \leftarrow 0$ ).

In step S207, the CPU 28 judges whether the input of  
20 the audio data transport stream that should be written in the AOB\_FILE# has ended. When the input of an audio data transport stream that has been encoded by the AAC encoder 25 and encrypted by the scrambling unit 26 into the RAM 22 continues and it is necessary to continue the writing  
25 of cluster data, the CPU 28 gives the judgement "No" in step S207 and the processing advances to step S209.





by the AAC encoder 25, the scrambling unit 26, and the modem apparatus 27 is complete, this means that the input of the audio data transport stream to be written into the AOB\_FILE#z will also be complete, so that the judgement  
5 "Yes" is given in step S207 and the processing advances to step S208. In step S208, the CPU 28 stores the value of the variable Data\_Size in the TKTMSRT of the TKI#z as the TMSRT\_Entry#y for the AOB\_ELEMENT#y. After storing the audio data accumulated in the RAM 22 in the AOB file  
10 corresponding to the AOB#z, the processing in this flowchart ends.

The above processing results in an encrypted audio data transport stream being stored in the flash memory card 31. The following procedure is then used to store the  
15 FileKey required for decrypting this encrypted audio data transport stream in the authentication region.

When the audio data transport stream has been inputted via input route RT1, the AOB file(s), the file storing the TKMG, the file storing the PLMG, and the encryption key  
20 storing file storing a different FileKey for each AOB are sent to the recording apparatus by a provider of the electronic music distribution service. The CPU 28 receives these files and writes the AOB file(s), the file storing the TKMG, and the file storing the PLMG into the user region  
25 of the flash memory card 31. On the other hand, the CPU 28 writes only the encryption key storing file storing a

different FileKey for each AOB into the authentication region.

When the audio is inputted via the input route RT2 or RT3, the CPU 28 generates a different FileKey every time the encoding of a new AOB commences and sets the generated key in the scrambling unit 26. In addition to being used by the scrambling unit 26 to encrypt the present AOB, this FileKey is stored following the FileKey Entry in the encryption key storing file present in the authentication region.

With the present embodiment describes above, the files storing AOBs are encrypted using different encryption keys, so that if the encryption key used to encrypt one file is decoded and exposed, the exposed encryption key can only be used to decrypt a file storing one AOB, with such exposure having no effect on other AOBs that are stored in other files. This minimizes the damage caused when one encryption key is exposed.

Note that while the above description focuses on an example system that is thought to be the most effective embodiment of the present invention, the invention is not limited to this system. Various modifications are possible within the scope of the invention, with examples of the such being given as (a) to (e) below.

(a) The above embodiment describes a semiconductor memory (flash memory card) as the recording medium used,

though the present invention can be applied to other media including optical discs, such as DVD-RAM, or a hard disk.

(b) In the above embodiment, the audio data was described as being in AAC format, though the present invention can also be applied to audio data in another format such as MP3 (MPEG 1 Audio Layer 3), Dolby-AC3, or DTS (Digital Theater System).

(c) While the file storing the TKMG and the file storing the PLMG were described as being received from the provider of the electronic music distribution service in a complete form, the main information used to create the TKMG and PLMG can be transmitted together with the encryption key storing file that stores a different encryption key for each AOB. The recording apparatus may then process this information to obtain the TKMG and PLMG which it then records in the flash memory card.

(d) For ease of explanation, the recording apparatus and playback apparatus were described as being separate devices, though a portable playback apparatus can be equipped with the functioning of the recording apparatus and a recording apparatus in the form of a personal computer can be equipped with the functions of the playback apparatus. Aside from the portable playback apparatus and personal computer recording apparatus, the functions of the playback apparatus and recording apparatus can also be provided to a communication device that is capable of downloading



content from a network.

As one example, a mobile telephone capable of Internet access may be provided with the functions of the playback apparatus and recording apparatus described in the above embodiment. This mobile telephone may store contents downloaded via a wireless network in the flash memory card 31 in the same way as in the above embodiment. Also, while the recording apparatus described in the above embodiment is provided with the modem apparatus 27 for connecting to the Internet, any other device capable of connecting to the Internet, such as a terminal adapter for an ISDN line, may be provided instead.

(e) The procedures shown in the flowcharts shown in FIGS. 55 to 58, FIG. 60, FIG. 63 to FIG. 65, and FIG. 68 can be achieved by executable programs that may be distributed and sold having been recorded on a recording medium. This recording medium may be an IC card, an optical disc, a floppy disk, or the like, with the programs recorded on the recording medium being used having first been installed into standard computer hardware. By performing processing in accordance with such installed programs, standard computer hardware can perform the same functioning as the playback apparatus and recording apparatus described in the above embodiment.

(f) While the above embodiment describes the case where a plurality of AOBs and a plurality of FileKeys are

stored on the flash memory card 31, only one AOB and one FileKey need be stored. Also, it is not essential for the AOBs to be encrypted, so that AOBs may be stored on the flash memory card 31 in ACC format.

5

## **SECOND EMBODIMENT**

The first embodiment only mentions the different storage regions in the flash memory card 31 and does not describe the internal hardware construction used. This second embodiment, however, describes the hardware construction of the flash memory card 31 in detail.

### **{69-1} Hardware Configuration of the Flash Memory Card 31**

FIG. 69 shows the hardware construction of the flash memory card 31. As shown in FIG. 69, the flash memory card 31 includes three IC chips, namely the control IC 302, the flash memory 303, and the ROM 304.

The ROM 304 includes the special region described in the first embodiment and is used to store the media ID mentioned in the first embodiment, in addition to a secure media ID 343 that is produced by encrypting the secure media ID.

The control IC 302 is a control circuit composed of active elements (logic gates) and includes an authorization unit 321, a command decoding unit 322, a master key storing unit 323, a special region access control unit 324, an

authentication region access control unit 325, a  
non-authentication region access control unit 326, and an  
encryption/decryption circuit 327.

The authorization unit 321 is a circuit that performs  
5 mutual authentication in challenge-response format with  
a device that tries to access the flash memory card 31.  
This authorization unit 321 includes a random number  
generator, an encrypter, and the like, and verifies that  
the device trying to access the flash memory card 31 is  
10 authentic by detecting whether the device includes the same  
encrypter as the authorization unit 321.

Here, mutual authentication in challenge-response  
format means that a first device sends challenge data to  
another device to check the authenticity of the other device.

15 The other device processes this challenge data in a  
predetermined way so as to prove its authenticity and sends  
the resulting data to the first device as response data.  
The first device compares the challenge data with the  
response data to judge whether the other device should be  
20 authenticated. Since this is mutual authentication, the  
processing is then repeated with the devices switching  
roles.

The command decoding unit 322 is a controller that  
includes a decoding circuit, a control circuit, and the  
25 like that interpret and execute a command (an instruction  
for the flash memory card 31) that has been inputted via

the COMMAND pin. The command decoding unit 322 controls the components 321-327 in the control IC 302 in accordance with the type of inputted command.

The commands issued to the flash memory card 31 include commands that read, write, or delete data in the flash memory 303. As examples of the commands related to the reading and writing of data, the commands "SecureRead address count" and "SecureWrite address count" access the authentication region, while the commands "Read address count" and "Write address count" access the non-authentication region. In these commands, the "address" is the number of the first sector to be accessed in the area subjected to the read (or write), while the "count" is the total number of sectors to be read (or written). In this case, a sector is the unit used for the reading and writing of data in the flash memory card 31, which is 512 bytes in the present example.

The master key storing unit 323 stores the master key 323a in an encrypted state in advance. The master key is the encryption key used to encrypt the media ID. When the flash memory card 31 is connected to a device, the master key 323a is passed over to the device in its encrypted form. The master key 323a is encrypted in a way that only allows decryption by a device that receives the master key using special key information (generally called a "device key").

The special region access control unit 324 is a circuit



the appropriate sector in the buffer memory is rewritten, and the block in the buffer memory is then written back into the flash memory 303.

The encryption/decryption circuit 327 performs encryption or decryption using the master key 323a stored in the master key storing unit 323 under the control of the authentication region access control unit 325 or the non-authentication region access control unit 326. When data is to be written into the flash memory 303, the encryption/decryption circuit 327 encrypts the data and writes it into the flash memory 303. Conversely, when data is to be read from the flash memory 303, the encryption/decryption circuit 327 decrypts the data. This encryption/decryption circuit 327 is provided to prevent users from performing unauthorized acts, such as dismantling the flash memory card 31 and directly analyzing the content of the flash memory 303 to obtain the passwords stored in the authentication region.

#### **{69\_70} Communication Sequence When Playing Back AOBs**

FIG. 70 shows the communication sequence performed when a playback apparatus connected to the flash memory card 31 reads the encryption key FileKey and plays back an AOB.

The playback apparatus issues a command to read the master key to the flash memory card 31 (sc1). Once this

command is issued, the command decoding unit 322 obtains the encrypted master key 323b that is stored in the master key storing unit 323 and passes it over to the playback apparatus (sc2).

5       The playback apparatus that receives the secure media ID uses the device key 211a that it stores itself to decrypt the secure media ID (sc3). The decryption algorithm used in the decrypting process corresponds to the encryption algorithm that was used when generating the encrypted master key 323b stored in the flash memory card 31, so that if  
10       the device key 211a used by the playback apparatus is a key whose use is expected (i.e., a proper key), the playback apparatus will be able to successfully obtain the master key by performing this decryption.

15       After receiving the master key, the playback apparatus issues a special command to the flash memory card 31 to read the media ID (sc4). The special region access control unit 324 obtains the media ID from the ROM 304 of the flash memory card 31 and passes it over to the playback apparatus  
20       (sc5). The encryption/decryption circuit 327 then encrypts the media ID using the master key obtained through the above decryption process (sc6). The algorithm used for this encryption is the same as the algorithm that was used to generate the secure media ID 343 stored in the flash  
25       memory card 31. As a result, a secure media ID that is the same as the secure media ID 343 of the flash memory

card 31 is obtained.

The playback apparatus that has succeeded in obtaining a secure media ID then performs mutual authentication with the authorization unit 321 of the flash memory card 31 (sc7).

5 This process results in both the playback apparatus and the authorization unit 321 having (a) information (OK/NG) showing whether the other device was successfully authenticated and (b) a time-variant secure key whose content depends on the authentication result.

10 When the mutual authentication has succeeded, the playback apparatus generates a command for accessing the authentication region of the flash memory card 31. As one example, when data is to be read from the authentication region, the playback apparatus encrypts the parameters (i.e., a 24-bit address "address" and an eight-bit data length "count") of the "SecureRead address count" command using the secure key (sc8), and links these parameters with the tag of this command (i.e., a 6-bit code showing that this command is a "SecureRead") to produce an encrypted command (sc9), which the playback apparatus sends to the flash memory card 31 (sc10).

On receiving this encrypted command, the flash memory card 31 identifies the type of command from the tag (sc11). In the present example, the flash memory card 31 identifies that the command is a "SecureRead" command for a read from the authentication region.



When a read command has been identified, the encryption/decryption circuit 327 decrypts the parameters included in the command using the secure key (sc12) obtained during the mutual authentication (sc13).

5       The algorithm used to decrypt the parameters corresponds to the encryption algorithm that was used by the playback apparatus when generating the encrypted command, so that when mutual authentication succeeded, which is to say, when the secure key in the flash memory  
10       card 31 matches the secure key in the playback apparatus, the parameters obtained by this decrypting will be the parameters used by the playback apparatus.

On receiving a command including valid parameters, the authentication region access control unit 325 accesses  
15       the sectors specified by the valid parameters and reads the encryption key FileKey stored in these sectors from the authentication region. The encryption/decryption circuit 327 encrypts the encryption key FileKey stored in the file "AOBSA1.KEY" in the authentication region (sc15)  
20       using the secure key (sc14) obtained during the mutual authentication. After this, the authentication region access control unit 325 sends the encryption key FileKey stored in the file "AOBSA1.KEY" in the authentication region to the playback apparatus (sc16).

25       The playback apparatus decrypts (sc18) the encryption key FileKey it has received using the secure key (sc17)

obtained during the mutual authentication. The decryption algorithm used here corresponds to the algorithm that was used by the flash memory card 31 to encrypt the encryption key FileKey, so that the original encryption key FileKey can be obtained. After this, the obtained encryption key FileKey is decrypted using the master key 323b and the media ID to obtain the encryption key FileKey (sc20).

Once the encryption key FileKey has been obtained and an AOB corresponding to this encryption key FileKey has been read from the non-authentication region (sc21), the AOB is decrypted using this encryption key FileKey and music is simultaneously played back.

#### **[69\_70\_71] Detailed Communication Sequence During Mutual Authentication**

FIG. 71 shows the communication sequence used during the mutual authentication shown in FIG. 70 in detail. In this example, the flash memory card 31 and the playback apparatus perform mutual authentication in challenge-response format.

The authorization unit 321 in the flash memory card 31 generates a random number to test the authenticity of the playback apparatus (sc30) and sends this random number to the playback apparatus as challenge data (sc50). In order to prove its own authenticity, the playback apparatus encrypts the challenge data (sc31) and sends the result





execution of the mutual authentication procedure is set as the condition for accessing the authentication region.

As one alternative, each side may produce the secure key by taking a logical XOR of the encrypted challenge data produced by this side, response data received from the other side, and the secure media ID.

The above embodiments have data that relates to the protection of copyrights stored in the authentication region and other data stored in the non-authentication region. This enables the realization of a semiconductor memory card capable of simultaneously storing both digital productions whose copyrights need to be protected and digital productions subject to no such restrictions.

Although the present invention has been fully described by way of example with reference to accompanying drawing, it is to be noted that various changes and modifications will be apparent to those skilled in the art. Therefore, unless such changes and modifications depart from scope of the present invention, they should be constructed as being included therein.



1 3. A semiconductor memory card according to Claim 2,  
2 wherein each audio track further includes  
3 (1) attribute information and  
4 (2) link information  
5 for each audio object included in the audio track,  
6 the attribute information showing a type, out of type  
7 (a), type (b), type (c) and type (d), for each audio object,  
8 type (a) being an entire audio track,  
9 type (b) being a first part of an audio track,  
10 type (c) being a middle part of an audio track, and  
11 type (d) being an end part of an audio track, and  
12 the link information for each audio object that is  
13 type (b) or type (c) showing which audio object follows  
14 the audio object.

1 4. A semiconductor memory card according to Claim 3,  
2 wherein the plurality of audio objects includes:  
3 at least one audio object that only contains valid  
4 data that needs to be played back; and  
5 at least one audio object that contains (1) valid  
6 data and (2) invalid data located at least one of before  
7 and after the valid data, the invalid data not needing to  
8 be played back,  
9 each audio track further including block information  
10 for each audio object in the audio track, the block  
11 information including:

12           an offset measured from the storage position of the  
13   corresponding audio object given in the management  
14   information; and

```

15         length information showing a length of the valid data
16 that starts from a position indicated by the offset,
17         the attribute information for an audio object showing
18 whether the valid data indicated by the offset and the length
19 information

```

20 (a) corresponds to an entire audio track,  
21 (b) corresponds to a first part of an audio track,  
22 (c) corresponds to a middle part of an audio track,  
23 or  
24 (d) corresponds to an end part of an audio track.

1 5. A semiconductor memory card according to Claim 4,  
2 wherein audio tracks can be played back according  
3 to standard playback or intermittent playback,  
4 standard playback being a mode where the valid data  
5 in the audio objects composing the audio tracks is played  
6 back without any valid data being omitted and  
7 intermittent playback being a mode where (1) omission  
8 of valid data equivalent to a first period and (2) playback  
9 of valid data equivalent to a second period, are repeated,  
10 each audio track further including a plurality of  
11 pieces of entry position information that show internal  
12 positions of the valid data within the audio object at



13 intervals that are equivalent to the first period,  
14 and the block information for an audio object showing:  
15 the offset that indicates a difference between (1)  
16 the internal position shown by a first piece of entry  
17 position information for the audio object and (2) the  
18 storage position for the audio object given in the  
19 management information; and  
20 a length of the valid data that starts at a position  
21 indicated by the offset.

1 6. A playback apparatus for a semiconductor memory card,  
2 the semiconductor memory card including (1) a  
3 protected area that can be accessed by a device connected  
4 to the semiconductor memory card only if the device has  
5 been found to be authentic, the protected area storing an  
6 encryption key sequence composed of a plurality of  
7 encryption keys arranged into a predetermined order, and  
8 (2) an unprotected area that can be accessed by any device  
9 connected to the semiconductor memory card, the unprotected  
10 area storing at least one audio track and management  
11 information, the at least one audio track including a  
12 plurality of encrypted audio objects, and the management  
13 information showing which encryption key, out of the  
14 plurality of encryption keys, corresponds to each audio  
15 object stored in the unprotected area,  
16 the playback apparatus comprising:







1 11. A computer-readable storage medium storing a program  
2 which, when executed by a computer, has the computer play  
3 back data from a semiconductor memory card,

4 the semiconductor memory card including (1) a  
5 protected area that can be accessed by a device connected  
6 to the semiconductor memory card only if the device has  
7 been found to be authentic, the protected area storing an  
8 encryption key sequence composed of a plurality of  
9 encryption keys arranged into a predetermined order, and  
10 (2) an unprotected area that can be accessed by any device  
11 connected to the semiconductor memory card, the unprotected  
12 area storing at least one audio track and management  
13 information, the at least one audio track including a  
14 plurality of encrypted audio objects, and the management  
15 information showing which encryption key, out of the  
16 plurality of encryption keys, corresponds to each audio  
17 object stored in the unprotected area,

18 the program comprising:

19 a reading step for reading one of the plurality of  
20 audio objects included in the at least one audio track from  
21 the semiconductor memory card and reading an encryption  
22 key that corresponds to the read audio object from the  
23 encryption key sequence stored in the protected area of  
24 the semiconductor memory card;

25 a decrypting step for decrypting the read audio object  
26 using the read encryption key; and

27 a playback step for playing back the decrypted audio  
28 object,  
29 wherein when the decrypting step has finished  
30 decrypting the read audio object, the reading step  
31 reads a different audio object included in an audio  
32 track,  
33 reads an encryption key corresponding to the different  
34 audio object from the encryption key sequence, and  
35 supplies the newly read encryption key to the  
36 decrypting step.

1 12. A computer-readable storage medium storing a program  
2 which, when executed by a computer, has the computer record  
3 data onto a semiconductor memory card,

4 the program including:

5 an encrypting step for assigning at least one of  
6 a plurality of encryption keys to each content included  
7 in the title, and encrypting each content using the  
8 encryption keys assigned to the contents to produce a  
9 plurality of audio objects;

10 a recording step for recording onto the semiconductor  
11 memory card the plurality of encryption keys as an  
12 encryption key sequence and the plurality of audio objects  
13 as at least one audio track.

1 13. A computer-readable storage medium according to Claim







3           the semiconductor memory card including (1) a  
4   protected area that can be accessed by a device connected  
5   to the semiconductor memory card only if the device has  
6   been found to be authentic, the protected area storing an  
7   encryption key sequence composed of a plurality of  
8   encryption keys arranged into a predetermined order and  
9   (2) an unprotected area that can be accessed by any device  
10   connected to the semiconductor memory card, the unprotected  
11   area storing at least one audio track and management  
12   information, the at least one audio track including a  
13   plurality of encrypted audio objects, and the management  
14   information showing which encryption key, out of the  
15   plurality of encryption keys, corresponds to each audio  
16   object stored in the unprotected area,

17           the playback method comprising:

18           a reading step for reading one of the plurality of  
19   audio objects included in the at least one audio track from  
20   the semiconductor memory card and reading an encryption  
21   key that corresponds to the read audio object from the  
22   encryption key sequence stored in the protected area of  
23   the semiconductor memory card;

24           a decrypting step for decrypting the read audio object  
25   using the read encryption key; and

26           a playback step for playing back the decrypted audio  
27   object,

28           wherein when the decrypting step has finished

```

29 decrypting the read audio object, the reading step
30     reads a different audio object included in an audio
31 track,
32     reads an encryption key corresponding to the different
33 audio object from the encryption key sequence, and
34     supplies the newly read encryption key to the
35 decrypting step.

```

1 17. A recording method for recording a title composed of  
2 a plurality of contents onto a semiconductor memory card,  
3 the recording method comprising:  
4 an encrypting step for assigning at least one of a  
5 plurality of encryption keys to each content included in  
6 the title, and encrypting each content using the encryption  
7 keys assigned to the contents to produce a plurality of  
8 audio objects; and  
9 a recording step for recording onto the semiconductor  
10 memory card the plurality of encryption keys as an  
11 encryption key sequence and the plurality of audio objects  
12 as at least one audio track.

1 18. A recording method according to Claim 17,  
2 wherein after recording the plurality of encryption  
3 keys and the plurality of audio objects, the recording step  
4 also records management information onto the semiconductor  
5 memory card, the management information showing, for each

6 audio object, correspondence between a region on the  
7 semiconductor memory card storing the audio object and a  
8 storage position of the encryption key corresponding to  
9 the audio object.

1 19. A recording method according to Claim 18,  
2 wherein for each audio object, the recording step  
3 also records attribute information and link information  
4 onto the semiconductor memory card,  
5 the attribute information for each audio object  
6 showing a type, out of type (a), type (b), type (c) and  
7 type (d),  
8 type (a) being an entire audio track,  
9 type (b) being a first part of an audio track,  
10 type (c) being a middle part of an audio track,  
11 and  
12 type (d) being an end part of an audio track,  
13 and  
14 the link information for each audio object that is  
15 type (b) or type (c) showing which audio object follows  
16 the audio object.

1 20. A recording method for recording data onto a  
2 semiconductor memory card, comprising  
3 a first generating step for successively generating  
4 audio frames from an input signal received from outside

5 the recording apparatus, an audio frame being a smallest  
6 amount of data that can be independently decoded;

```

7         a writing step for creating a file on the
8 semiconductor memory card and writing the successively
9 generated audio frames into the file;

```

```
10         a second generating step for generating, whenever
11 the writing step has written a predetermined number of audio
12 frames into a file, a piece of entry information showing
13 a data length of an audio element that is composed of the
14 audio frames written into the file,
```

15            wherein whenever the second generating step has  
16 generated a predetermined number of pieces of entry  
17 information, the writing step creates a new file and writes  
18 the audio frames successively generated thereafter into  
19 the new file.

## ABSTRACT OF THE DISCLOSURE

09500601.052500  
003250.10903560

An audio stream is divided into a plurality of audio  
object (AOB) files that are recorded having each been  
5 encrypted using a different encryption key. At least one  
piece of track management information (TKI) is provided  
corresponding to each track. Playlist information (PLI)  
assigns a playback position in a playback order to each  
track when a plurality of tracks are to be played back one  
10 after the other.



FIG. 2

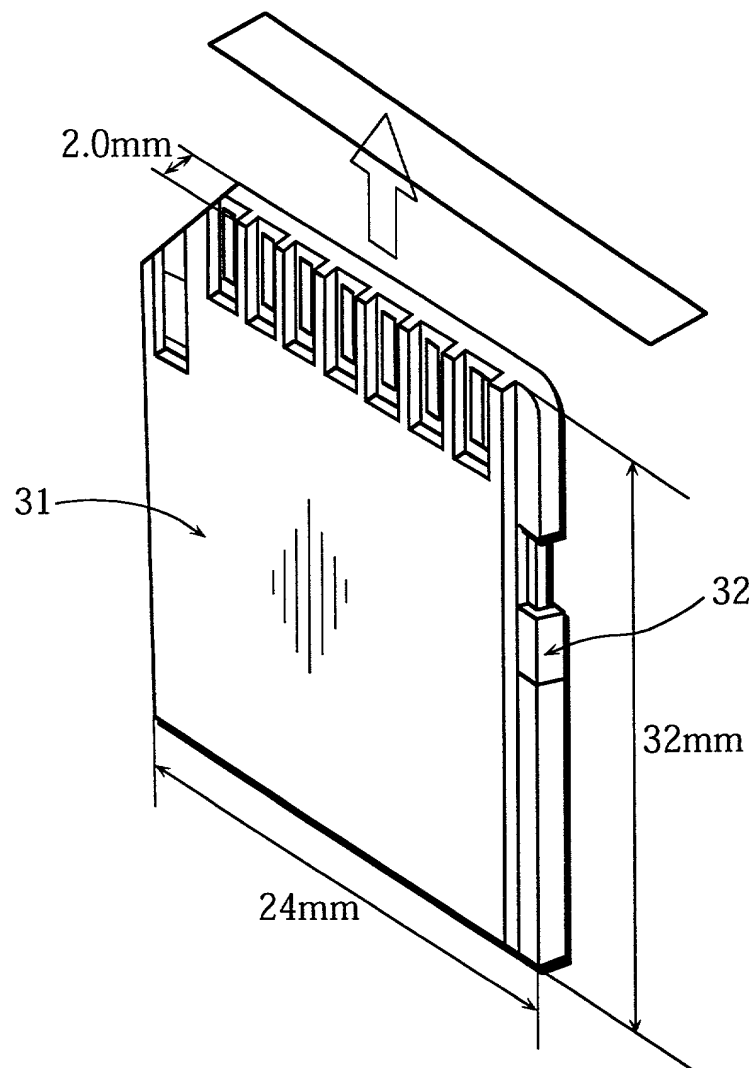






FIG. 4A

PHYSICAL LAYER

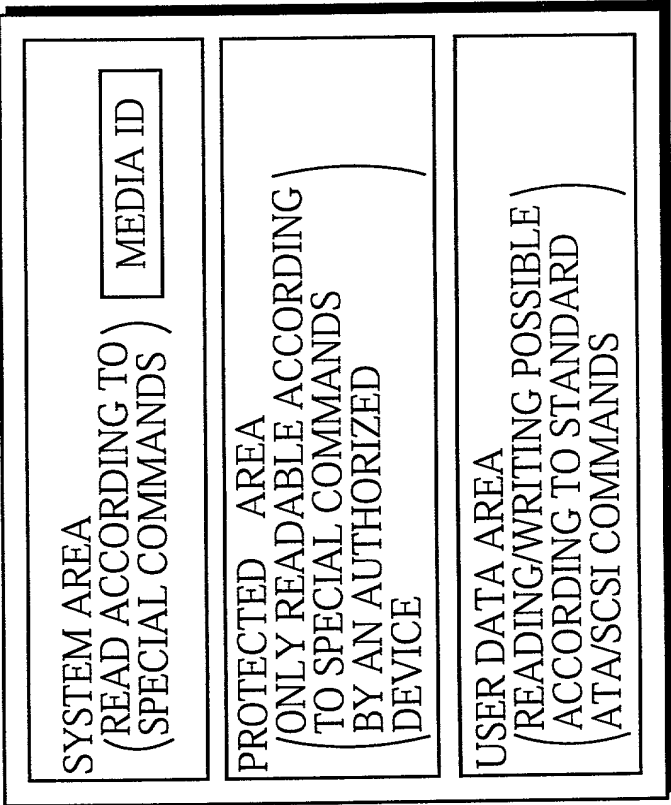


FIG. 4B FILE SYSTEM LAYER  
(ISO/IEC 9293)

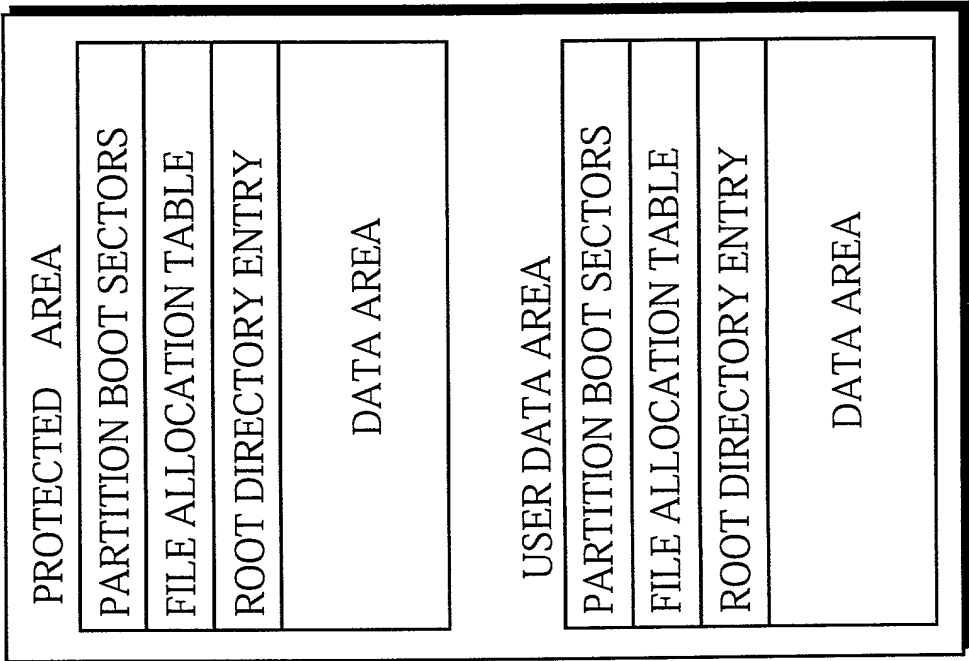


FIG. 5

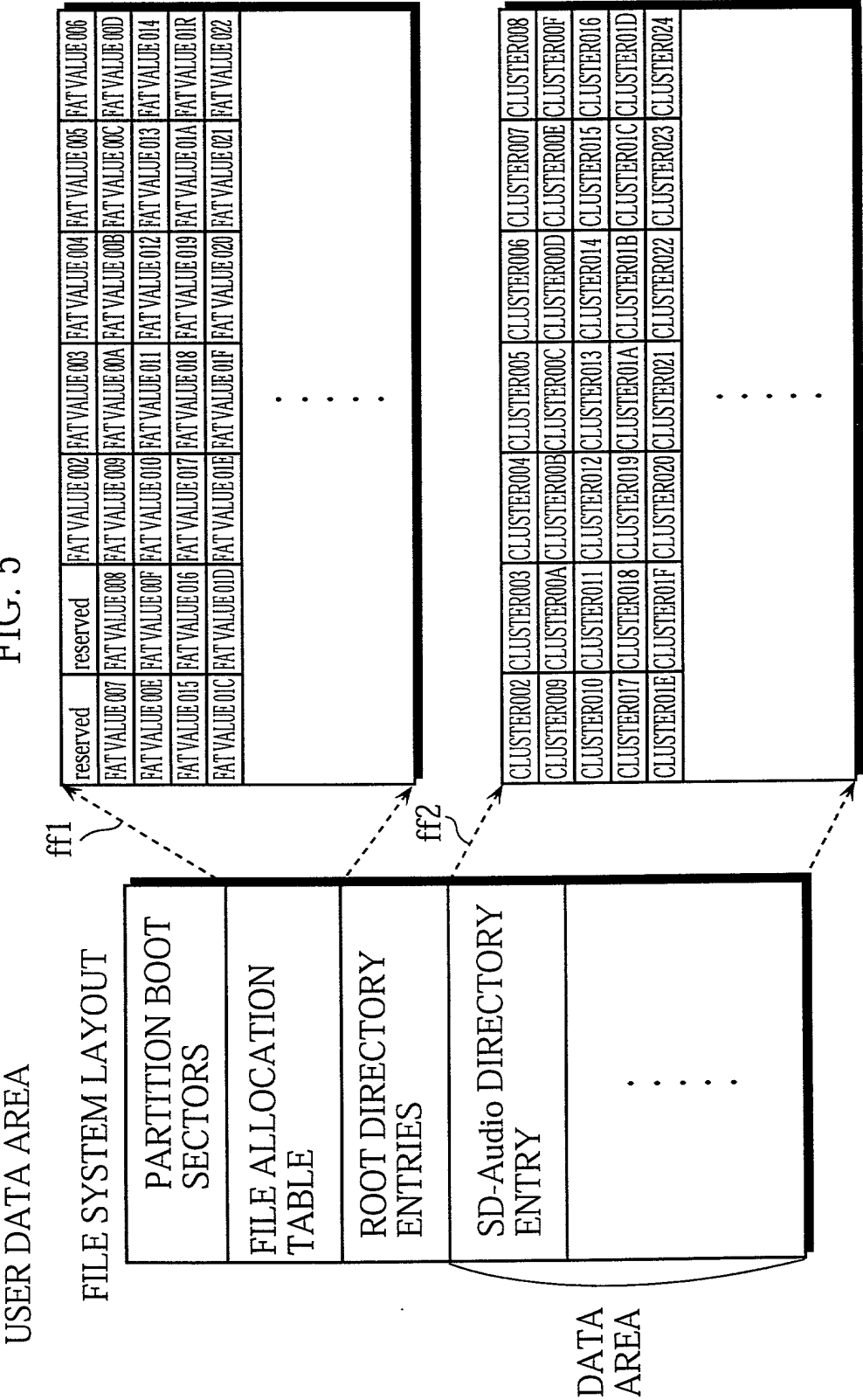
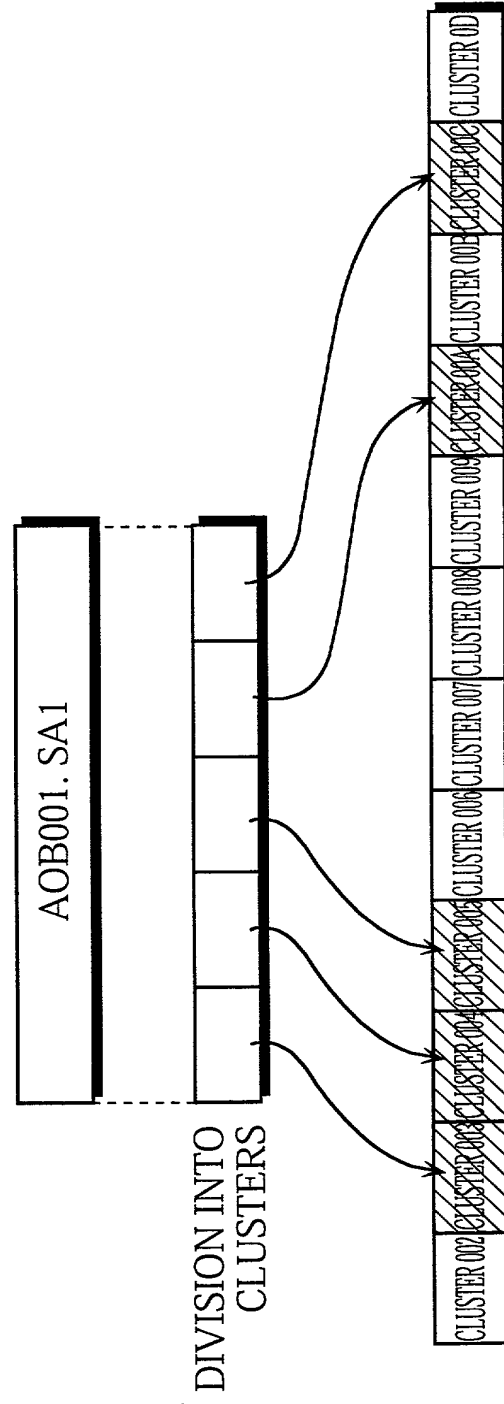


FIG. 6



SD-Audio DIRECTORY ENTRY

FIG. 7

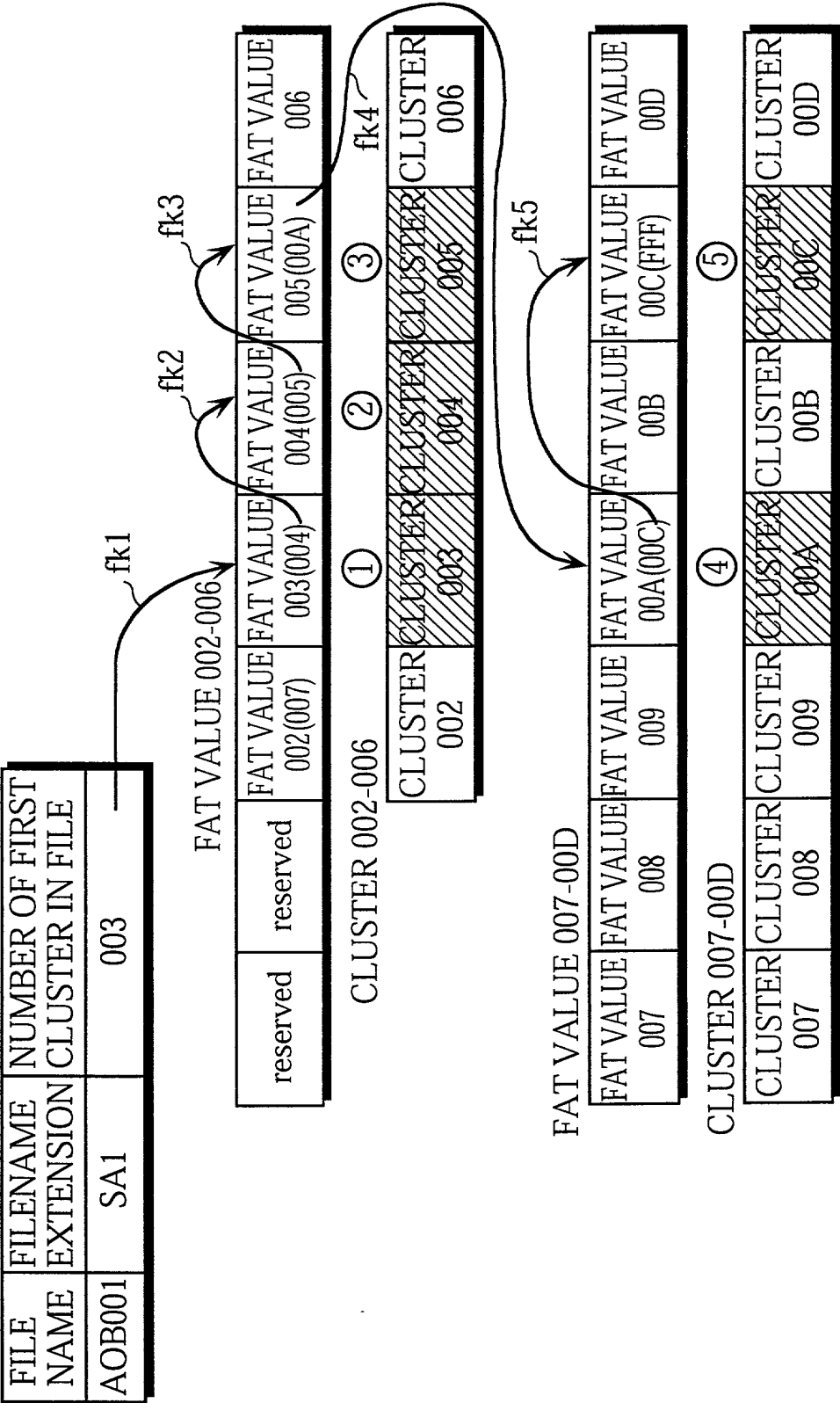


FIG. 8A

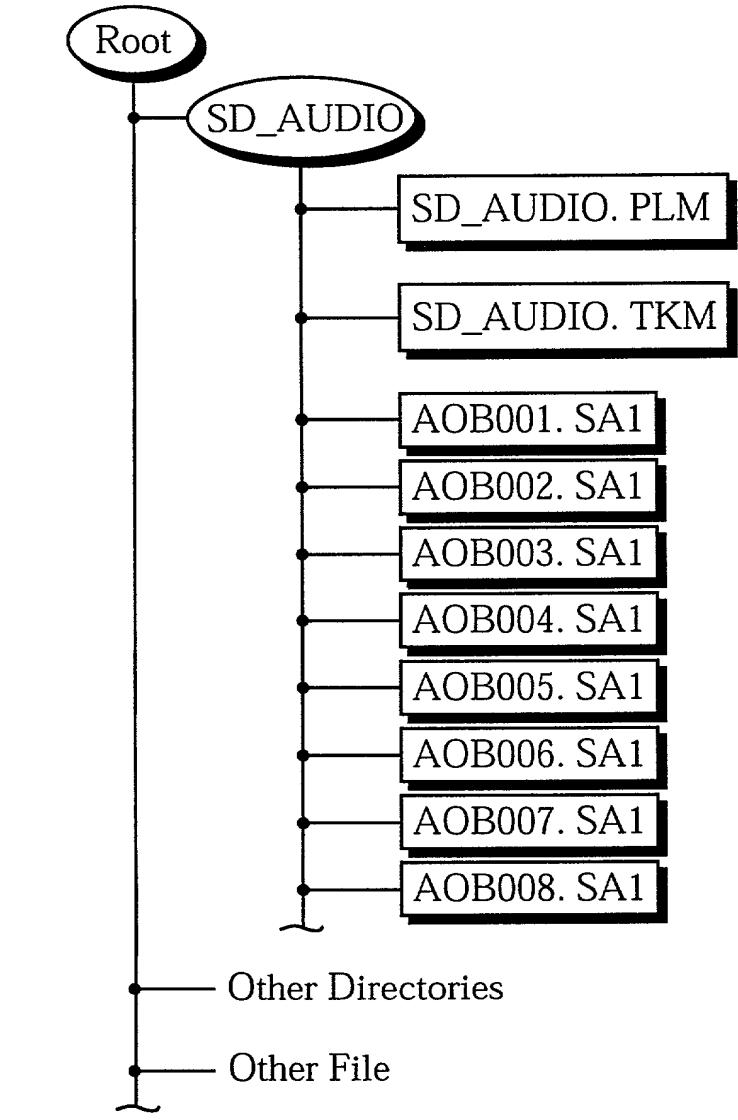


FIG. 8B

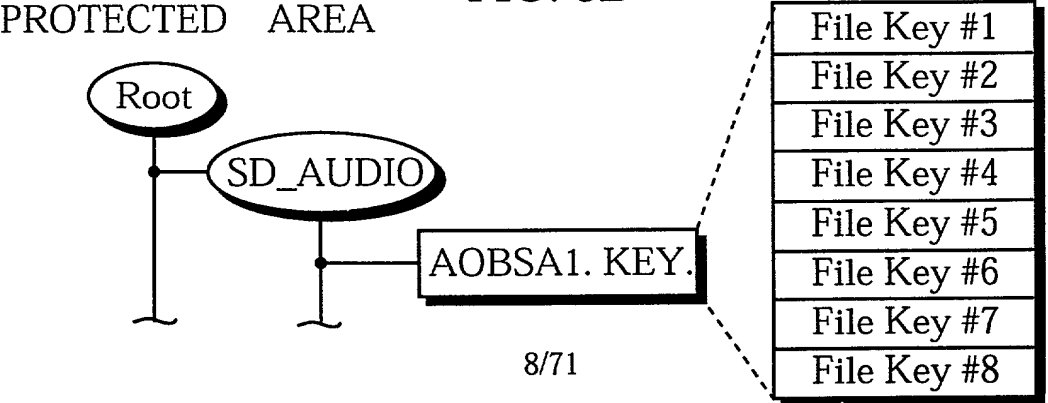




FIG. 10

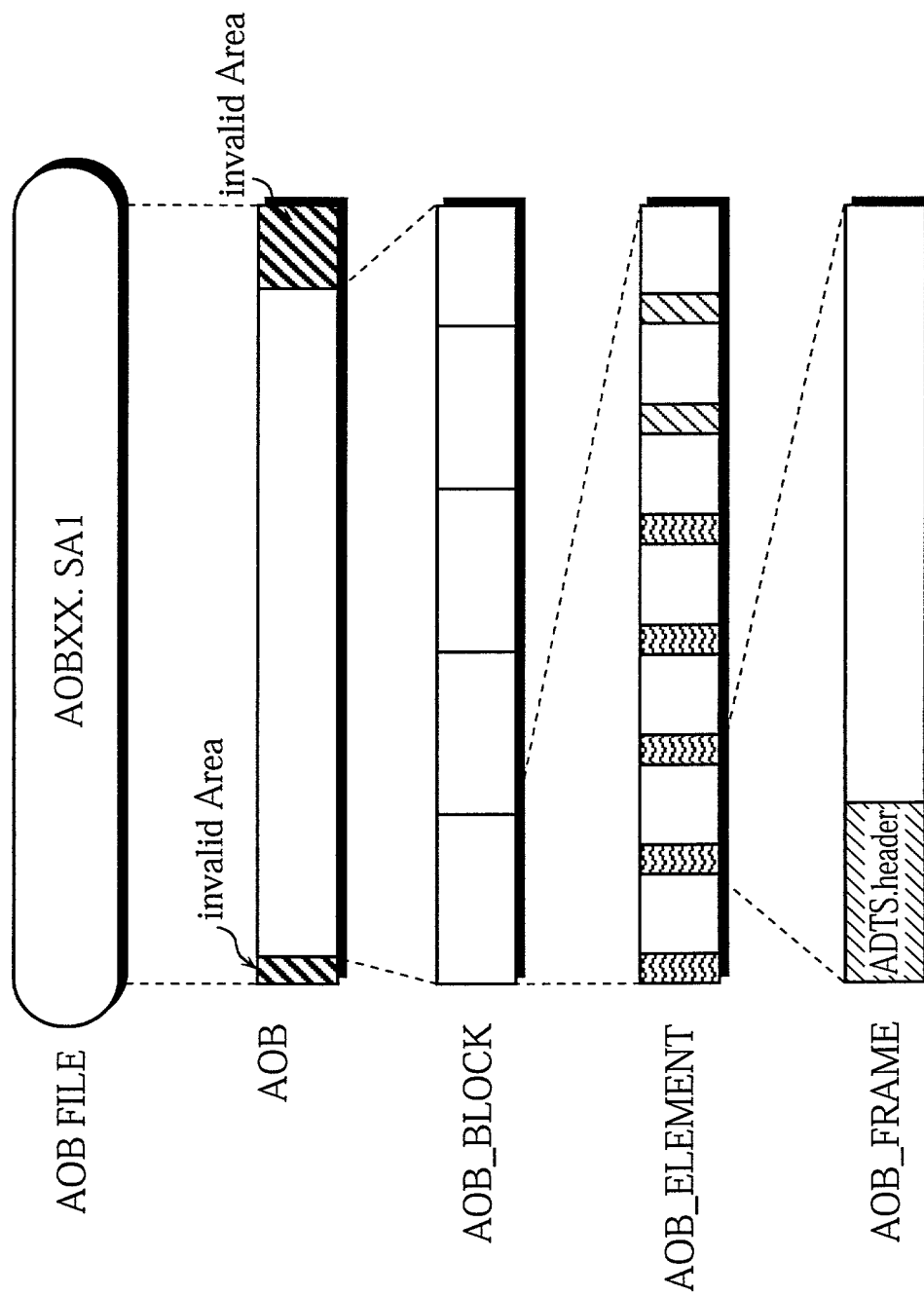


FIG. 11A

MPEG2-AAC format

format	Audio_Data_Transport_Stream(ADTS)
Profile	Low Complexity(LC) profile(mandatory)
bitrate per channel**	between 16kbit/s(min.)and 72kbit/s(max.) Other bitrates are optional.
sampling_frequency	48 kHz(mandatory) 44.1 kHz(mandatory) 32 kHz(mandatory) 24 kHz(mandatory) 22.05 kHz(mandatory) 16 kHz(mandatory)
channel_configuration	single_channel_element(mandatory) channel_pair_element(mandatory)
number_of_data_blocks_in_frame	1 header/1 raw_data_block(mandatory)

FIG. 11B

MPEG layer 3 format

format	MPEG-1 layer 3 MPEG-2 layer 3 low sampling frequency
bitrate per channel**	MPEG 1:between 16kbit/s and 96kbit/s MPEG 2 LSF:between 16kbit/s and 80kbit/s Other bitrates and variable bitrate are optional. Bitrate index "0000",i.e. "free format" is not supported.
sampling_frequency	48 kHz 44.1 kHz 32 kHz 24 kHz 22.05 kHz 16 kHz
mode	stereo joint_stereo single_channel

FIG. 11C

Windows Media Audio format

format	Windows Media Audio format
bitrate per channel**	between 8kbit/s and 80kbit/s Other bitrates are optional.
sampling_frequency	48 kHz 44.1 kHz 32 kHz 22.05 kHz 16 kHz
mode	monaural stereo



FIG. 12

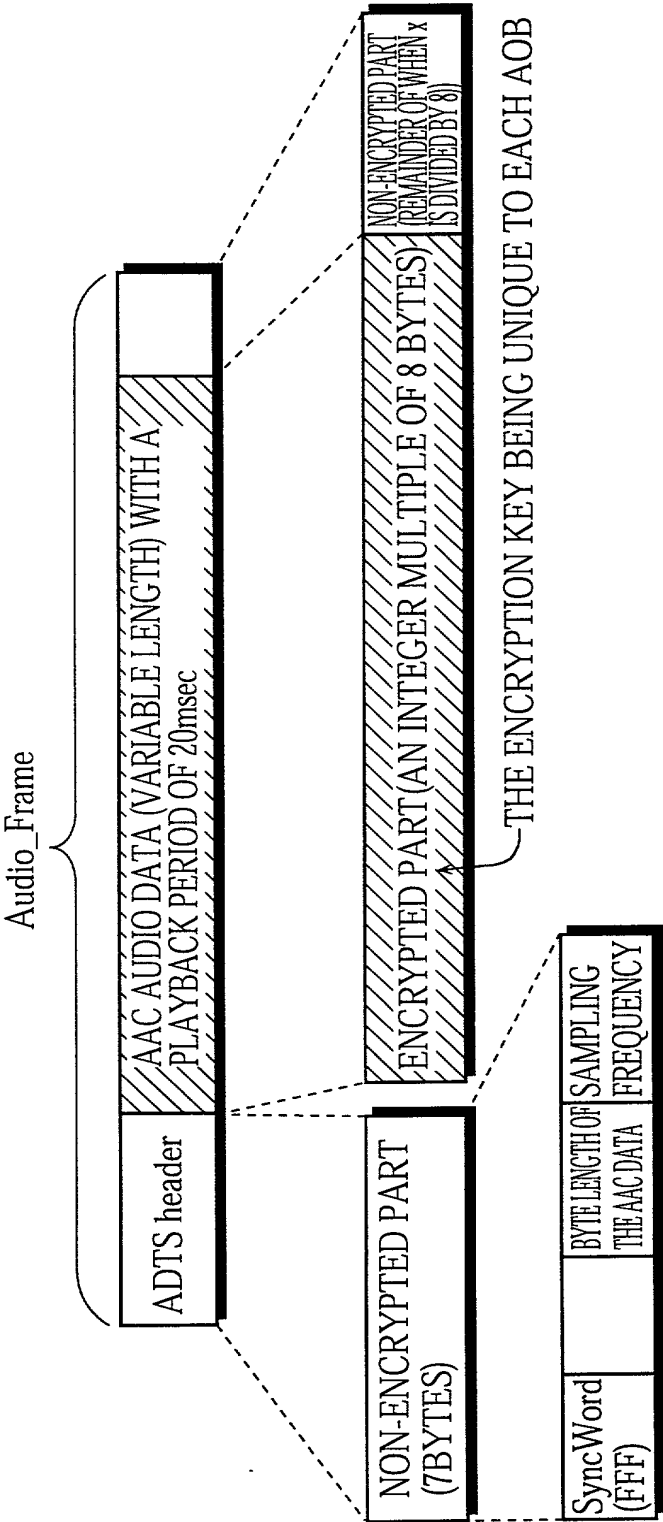
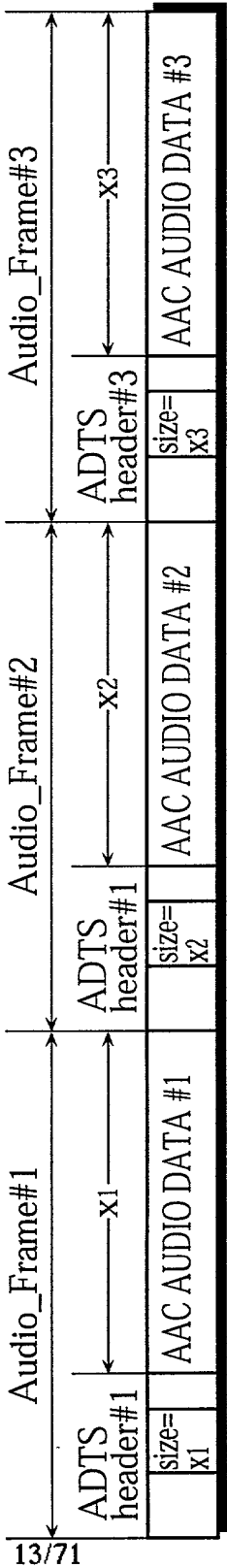


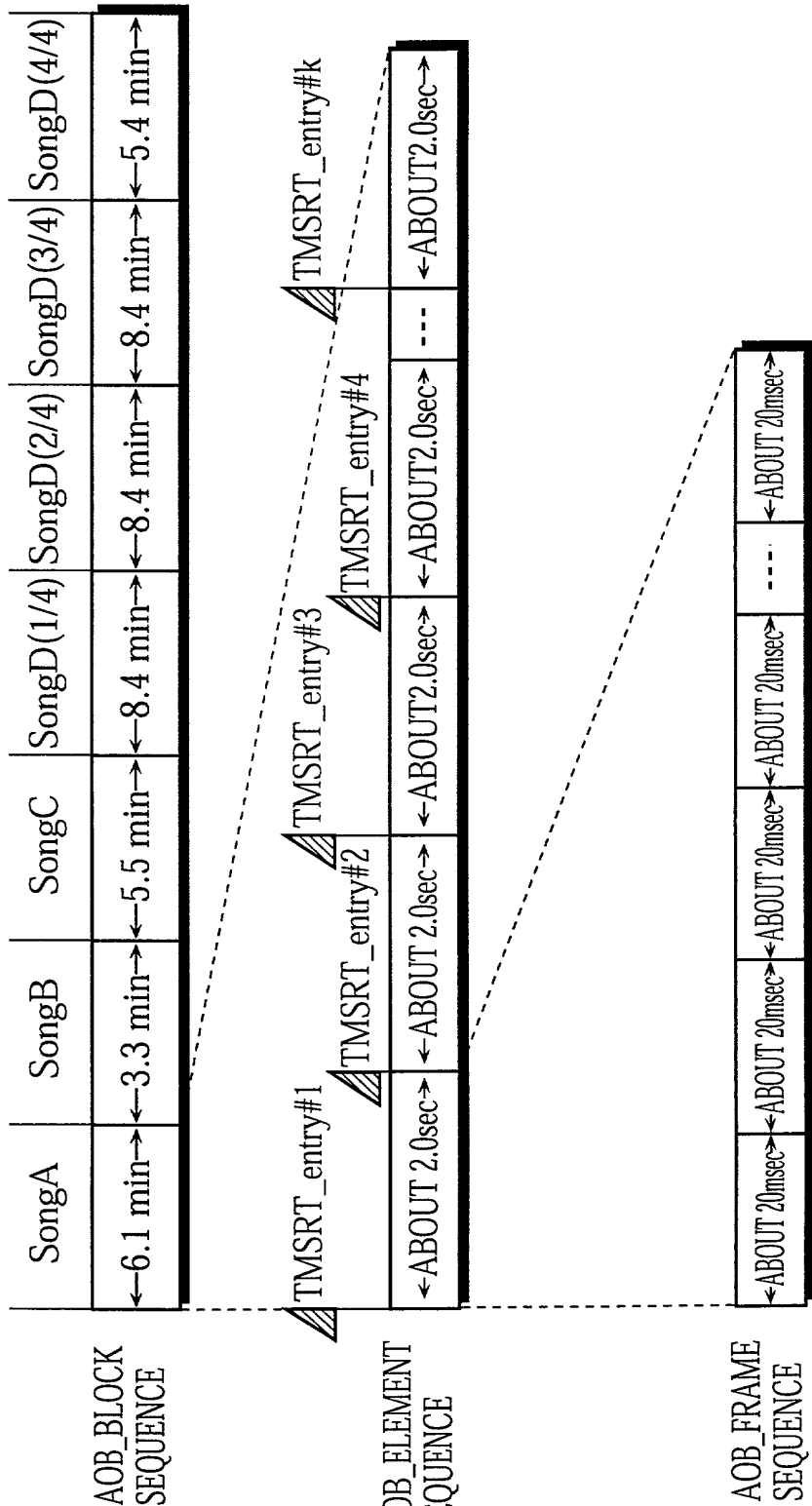
FIG. 13



(x1 ≠ x2 ≠ x3)



FIG. 15



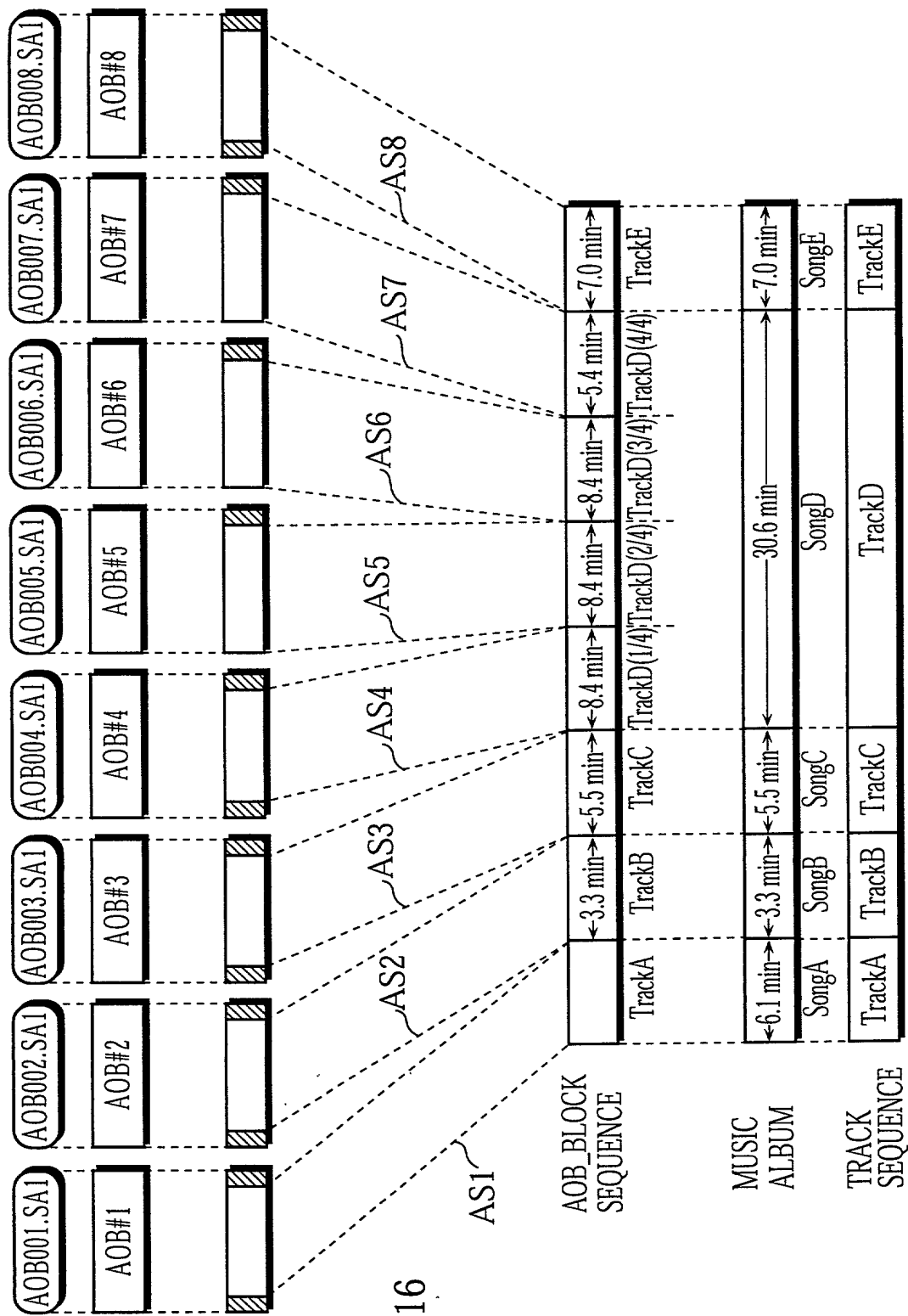


FIG. 16

FIG. 17

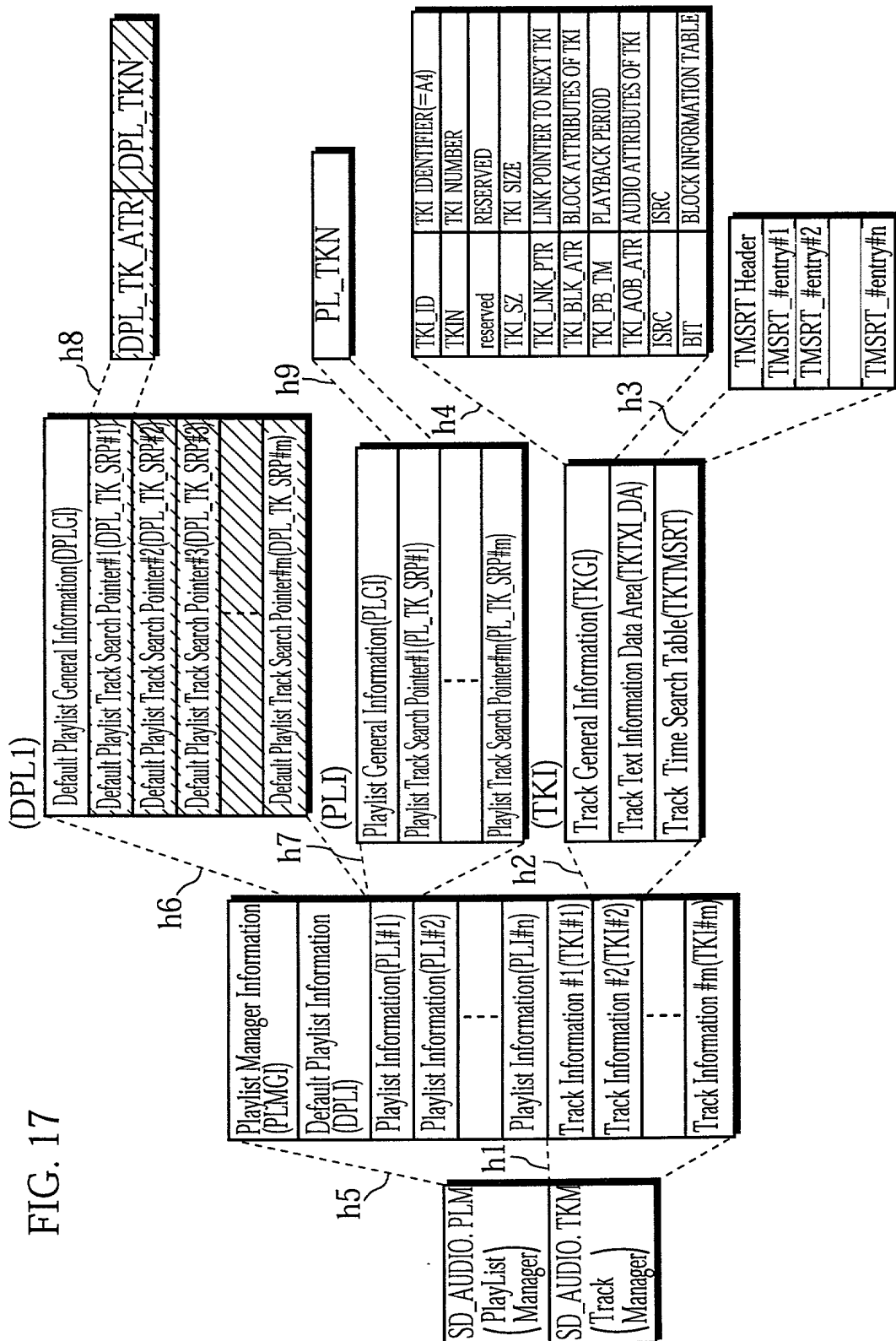


FIG. 18

(PLMG)

Playlist Manager Information(PLMGI)	FIXED LENGTH (2.5KByte)
Default Playlist Information(DPLI)	
Playlist Information#1 (PLI#1)	FIXED LENGTH (512Byte)
⋮	
Playlist Information#n(PLI#n) ( $1 \leq n \leq 99$ )	FIXED LENGTH (512Byte)
Track Information #1(TKI#1)	FIXED LENGTH (1024Byte)
⋮	
Track Information #m(TKI#m)	FIXED LENGTH (1024Byte)

$(1 \leq m \leq 999)$





FIG. 20

(DESCRIPTION ORDER)				
RELATIVE BYTE POSITION	FIELD NAME	CONTENT	NUMBER OF BYTES	
0 to 1	TMSRT ID	IDENTIFIER OF TMSRT	2 BYTES	
2 to 3	reserved	RESERVED	2 BYTES	
4 to 7	Total TMSRT_entry Number	TOTAL NUMBER OF TMSRT_entries	4 BYTES	
TOTAL			8 BYTES	

RELATIVE BYTE POSITION	FIELD NAME	CONTENT	NUMBER OF BYTES	
0 to 3	TMSRT_ENT	Size of AOB_ELEMENT	2 Byte × 252	
TOTAL			TOTAL 504 BYTES	

TMSRT Header

TMSRT\_entry #0

TMSRT\_entry #1

...

TMSRT\_entry #n

FIG. 21

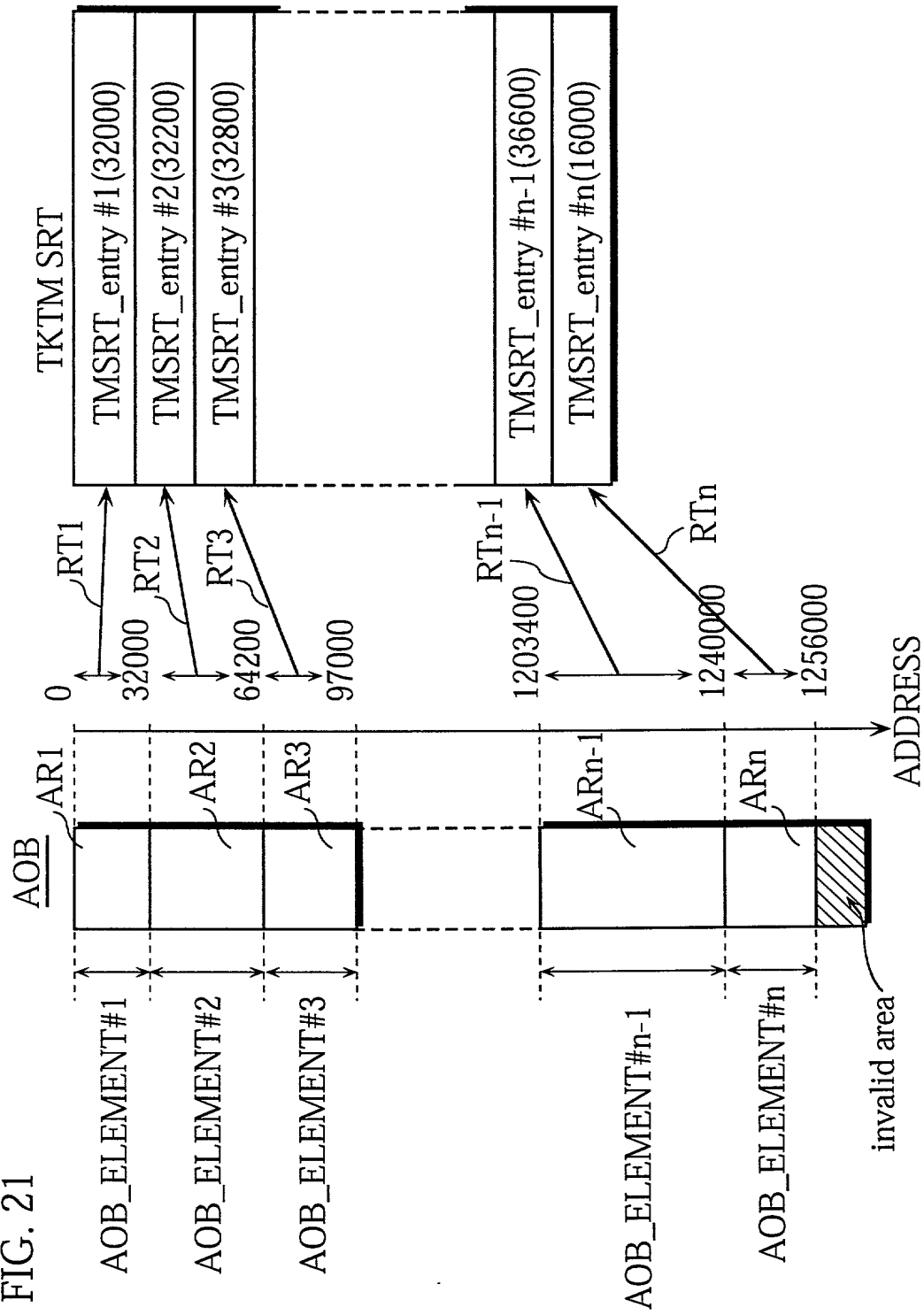


FIG. 22

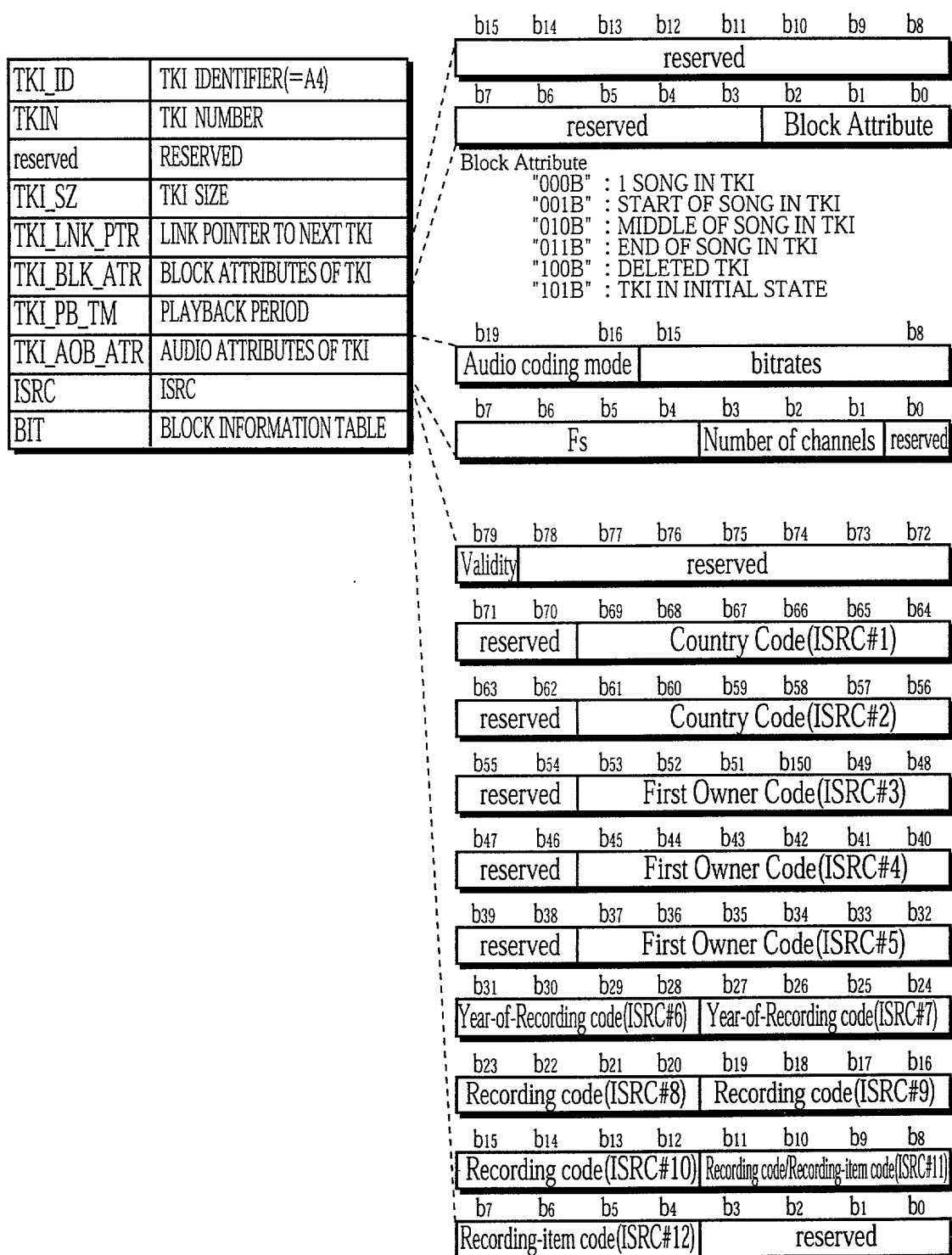


FIG. 23A

RELATIVE BYTE POSITION	FIELD NAME	CONTENT	NUMBER OF BYTES
60 to 63	DATA_OFFSET	FIRST ADDRESS OF AOB_BLOCK	4BYTES
64 to 67	SZ_DATA	DATA LENGTH OF AOB_BLOCK	4BYTES
68 to 71	TMSRTE_Ns	NUMBER OF TMSRT_entries	4BYTES
72 to 73	FNs_1st_TMSRTE	NUMBER OF AOB_FRAMES IN FIRST AOB_ELEMENT	2BYTES
74 to 75	FNs_Last_TMSRTE	NUMBER OF AOB_FRAMES IN LAST AOB_ELEMENT	2BYTES
76 to 77	FNs_Middle_TMSRTE	NUMBER OF AOB_FRAMES IN AOB_ELEMENTS	2BYTES
78 to 79	TIME_LENGTH	PLAYBACK PERIOD OF AOB_ELEMENTS	2BYTES
TOTAL			20BYTES

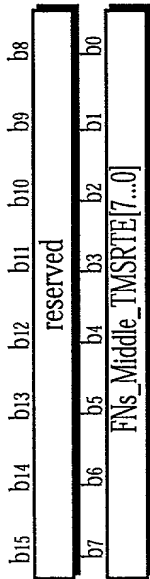


FIG. 23B

sampling frequency	FNs_Middle_TMSRTE	
	AAC	MPEGlayer3
48kHz	47*N	42*N
44.1kHz	43*N	38*N
32kHz	32*N	28*N
24kHz	24*N	42*N
22.05kHz	22*N	38*N
16kHz	16*N	28*N

※ N BEING THE PLAYBACK PERIOD "TIME\_LENGTH"  
OF AN AOB\_ELEMENT TO AN ACCURACY OF 1/1000TH  
OF ONE SECOND

FIG. 23C

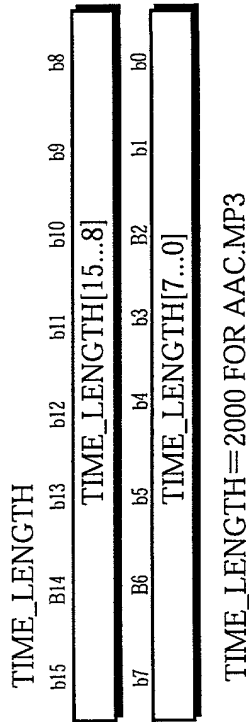






FIG. 26A

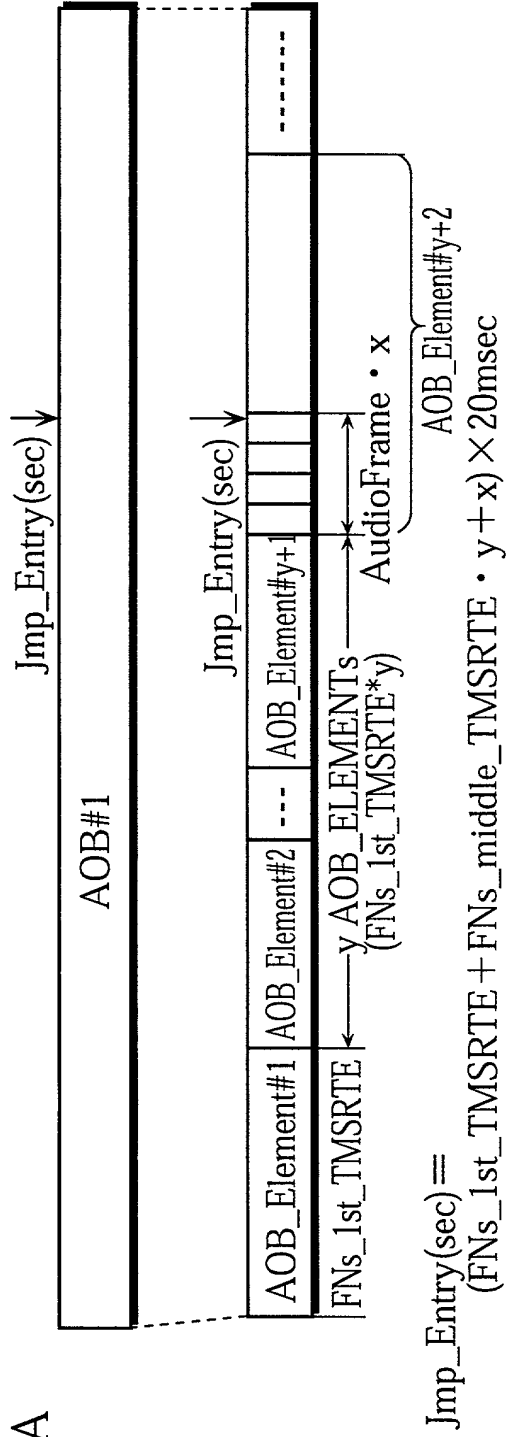


FIG. 26B

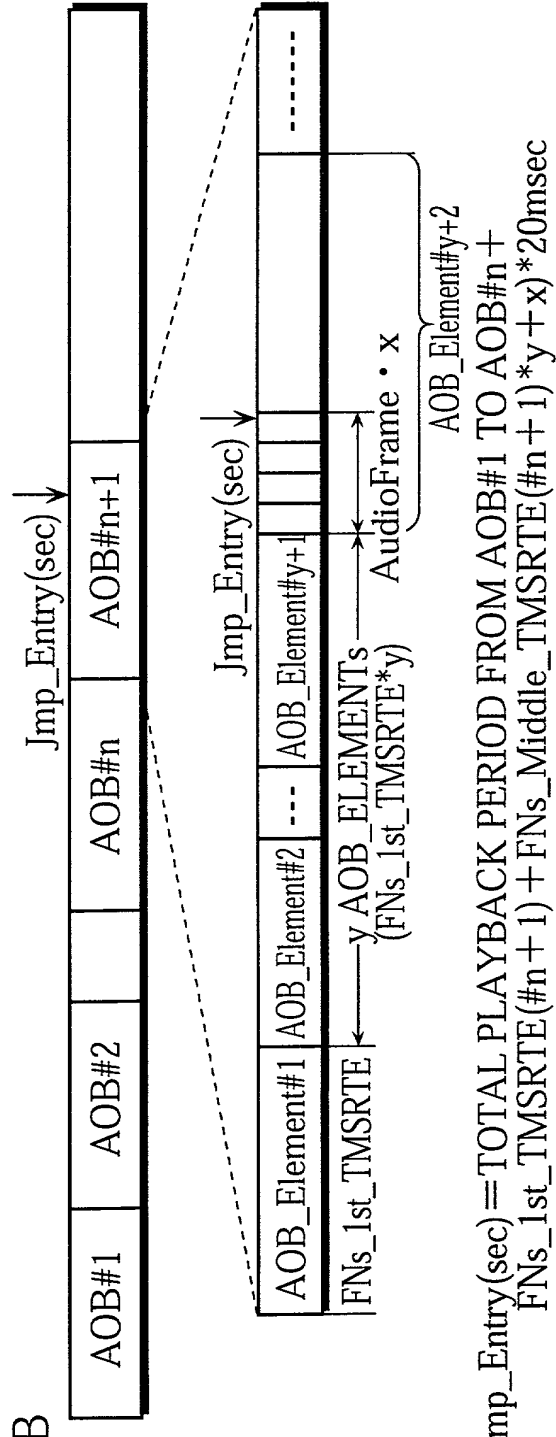


FIG. 27A

## DELETION OF TRACK B(case 1)

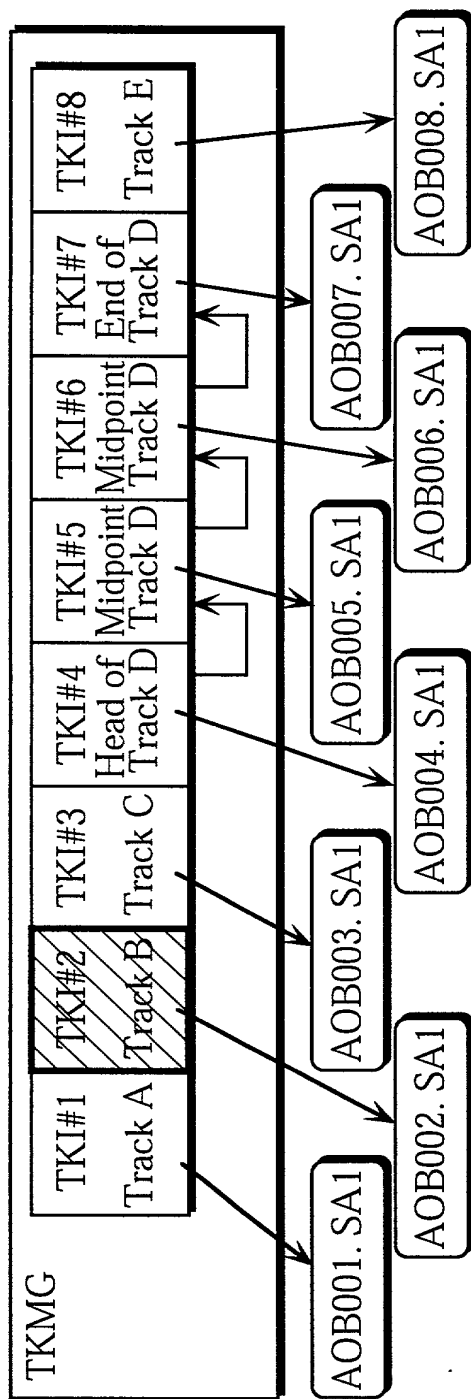


FIG. 27B

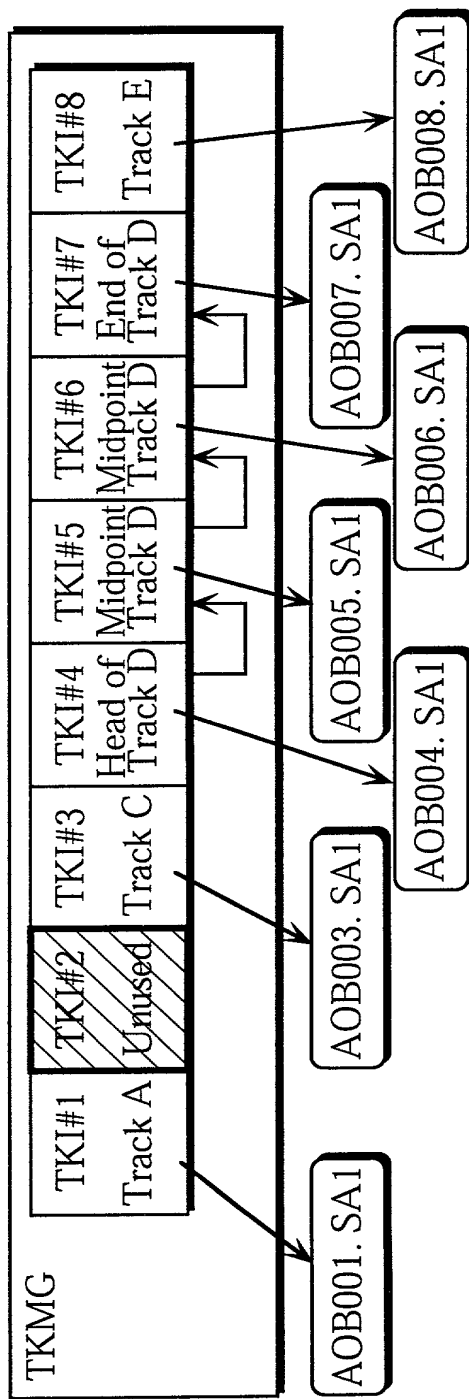




FIG. 28A UPDATING TrackManager USING UNUSED TKI AT DIFFERENT POSITIONS(case 2)

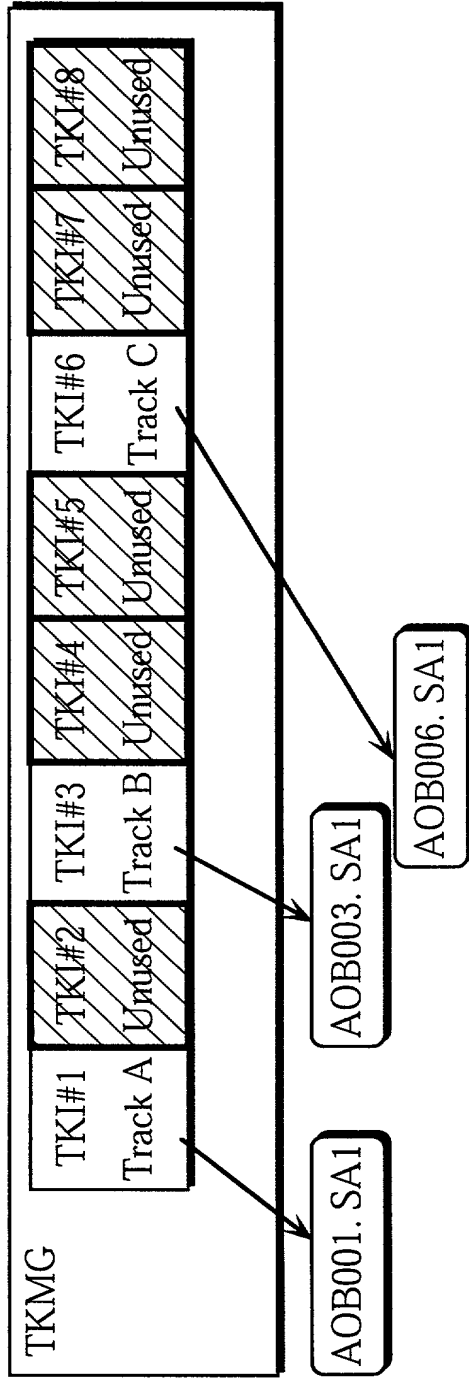


FIG. 28B

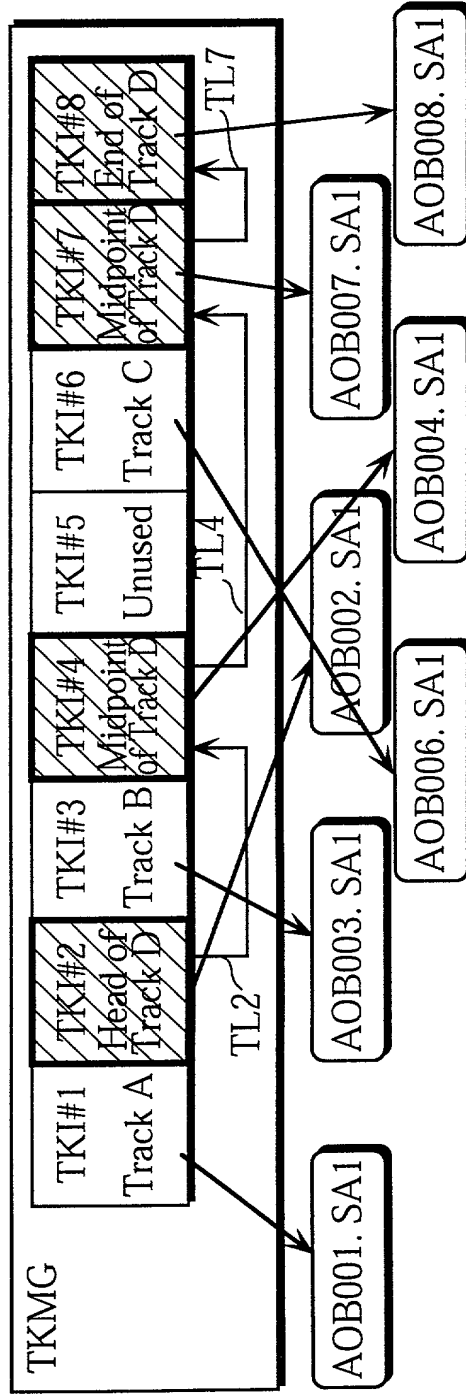


FIG. 29A

COMBINING OF Track C AND Track E TO PRODUCE NEW Track C (case 3)

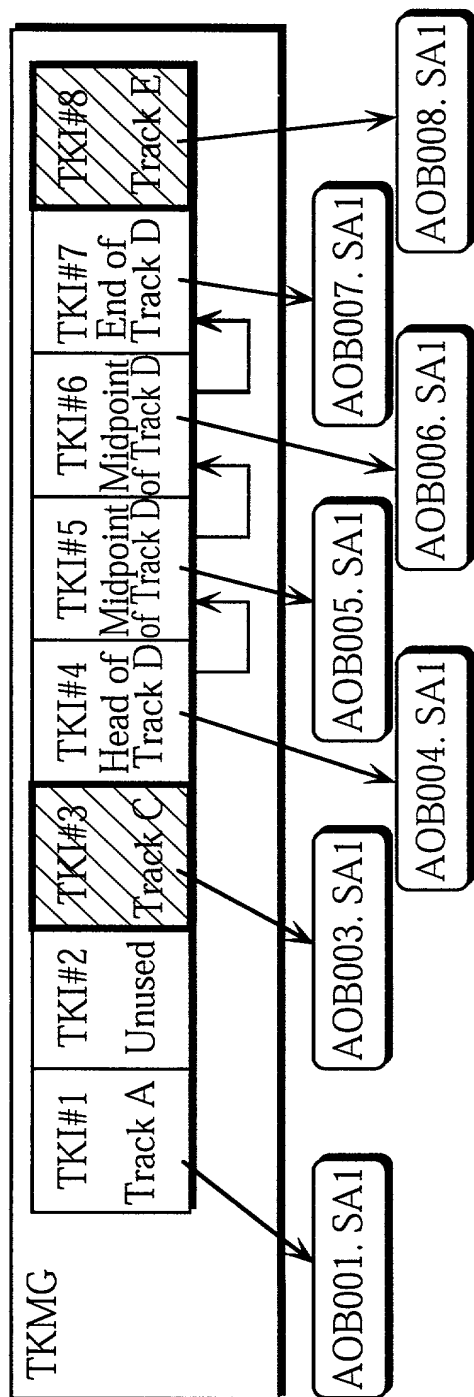


FIG. 29B

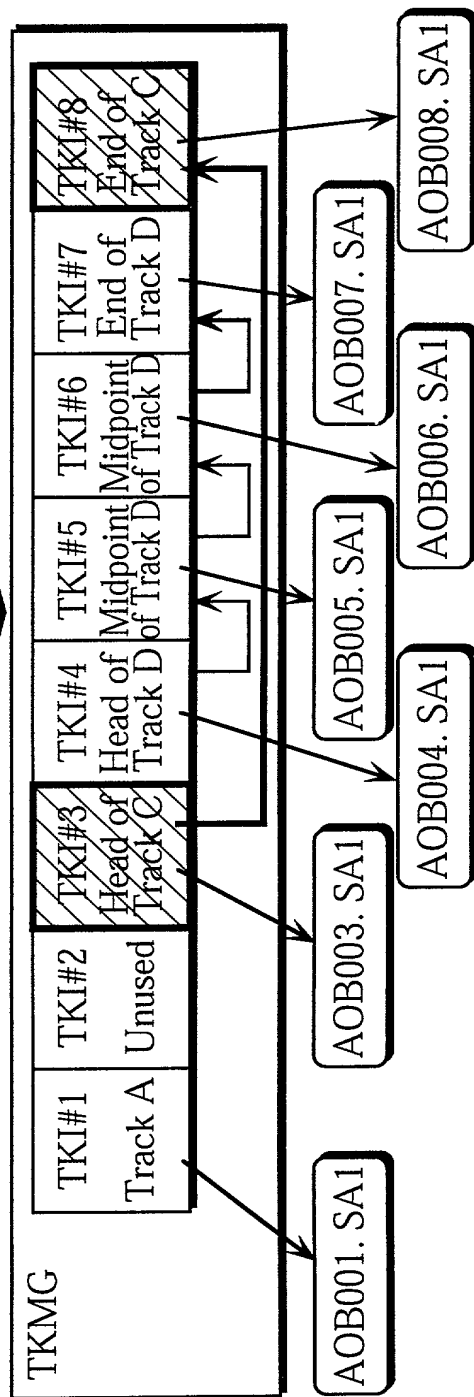


FIG. 30A

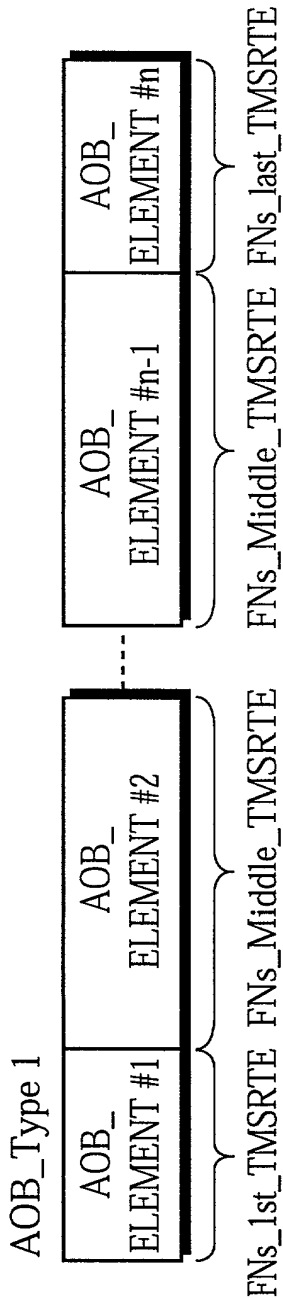


FIG. 30B

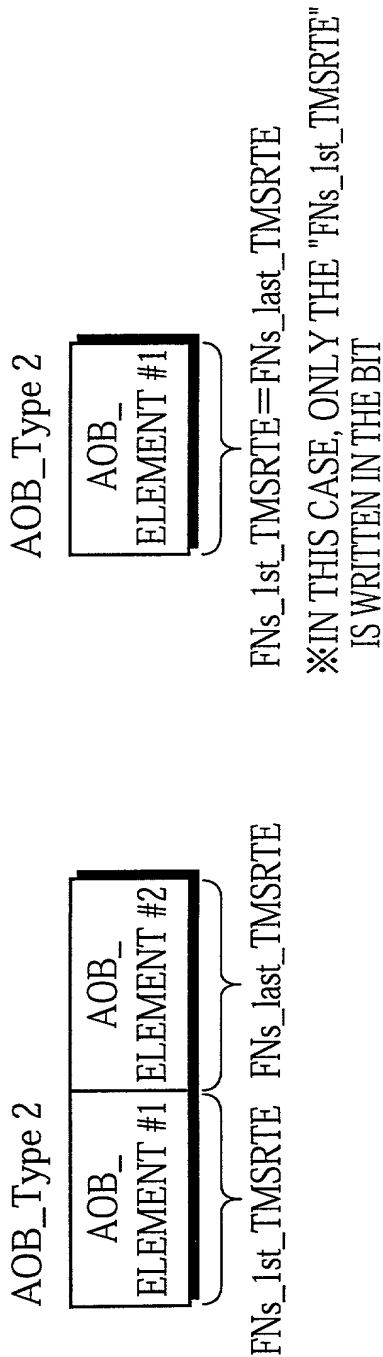




FIG. 32A

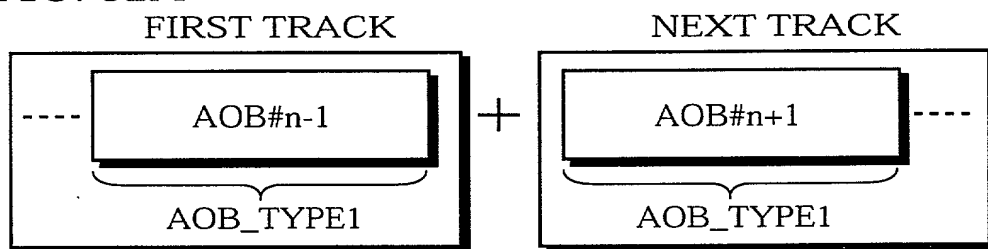


FIG. 32B

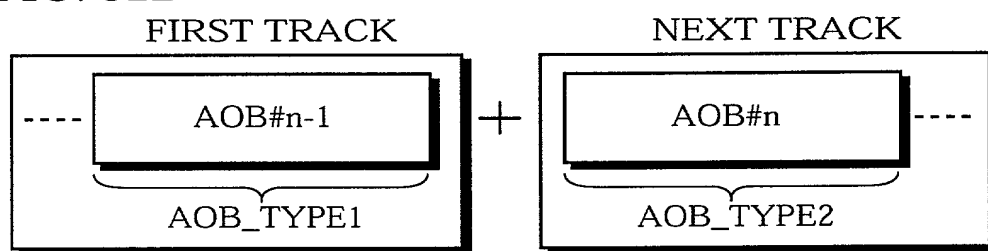


FIG. 32C

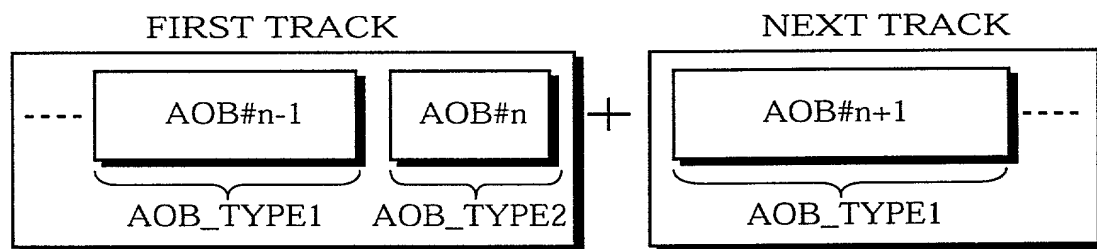


FIG. 32D

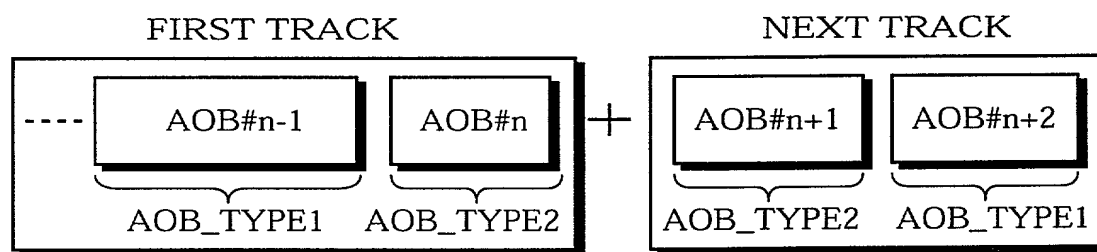


FIG. 32E

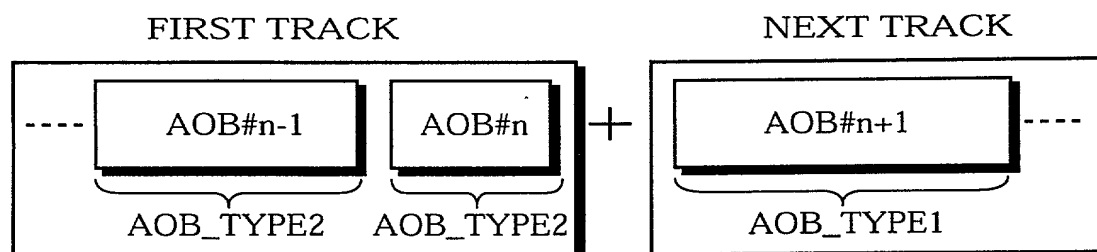


FIG. 33A

FIG. 33B

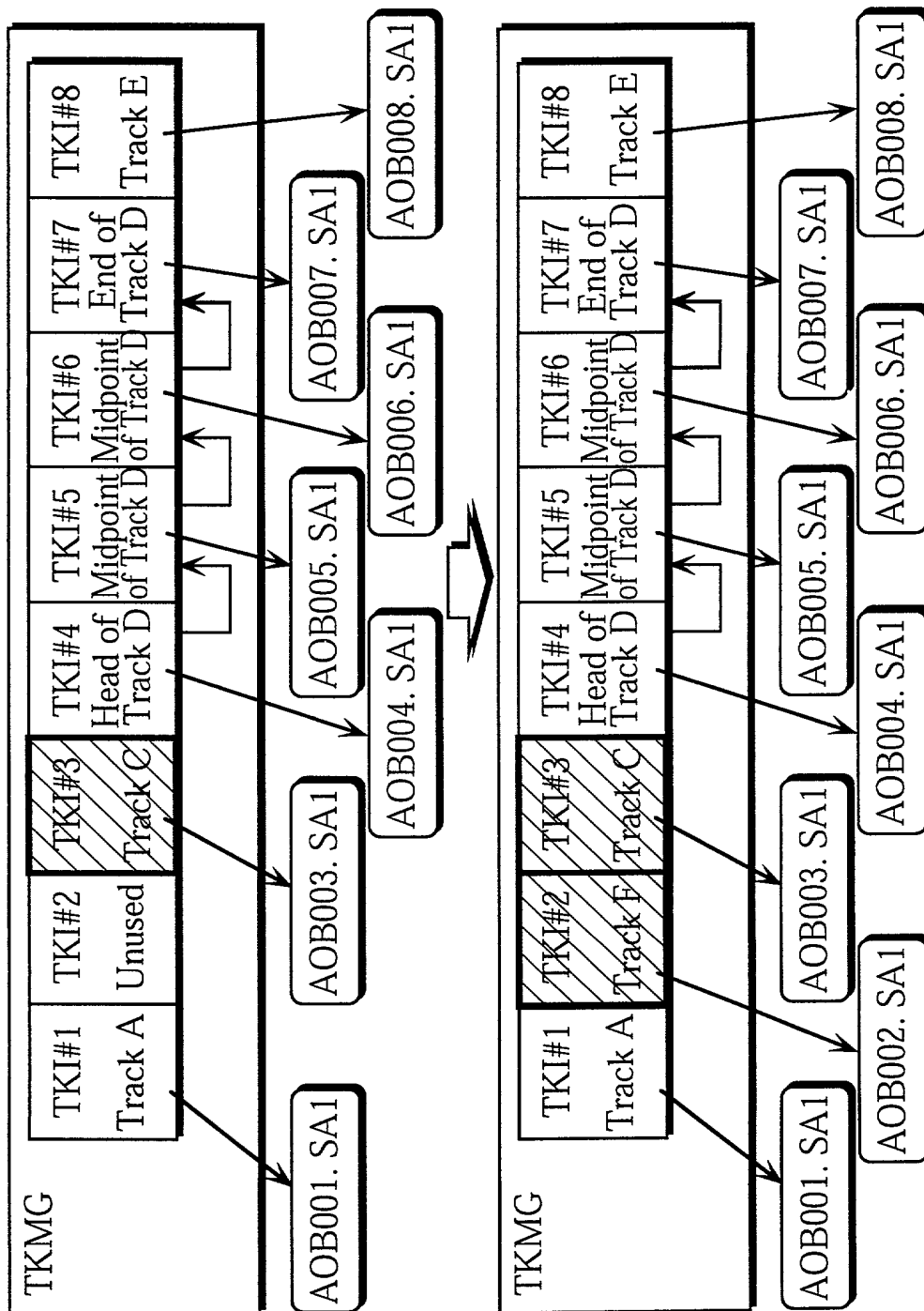


FIG. 34A

SD-Audio DIRECTORY ENTRY

FILE NAME	FILE NAME EXTENSION	FIRST CLUSTER NUMBER IN FILE
AOB003	SA 1	007

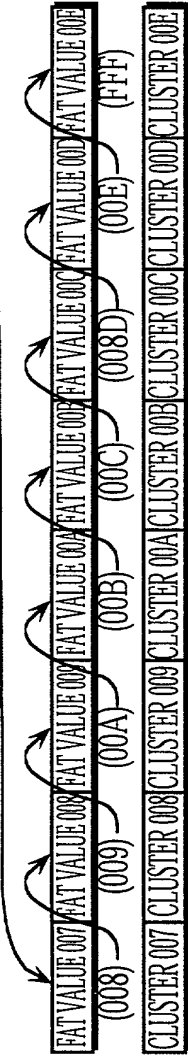


FIG. 34B

SD-Audio DIRECTORY ENTRIES

FILE NAME	FILE NAME EXTENSION	FIRST CLUSTER NUMBER IN FILE
AOB002	SA 1	00F
FILE NAME	FILE NAME EXTENSION	FIRST CLUSTER NUMBER IN FILE
AOB003	SA 1	007

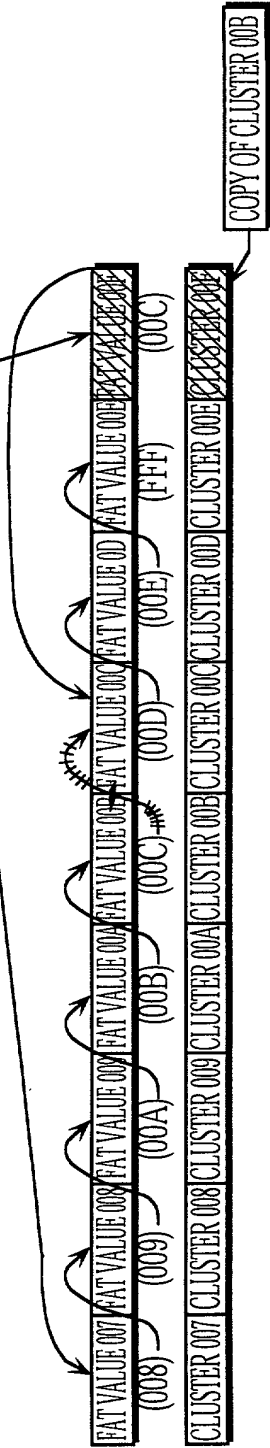


FIG. 35A

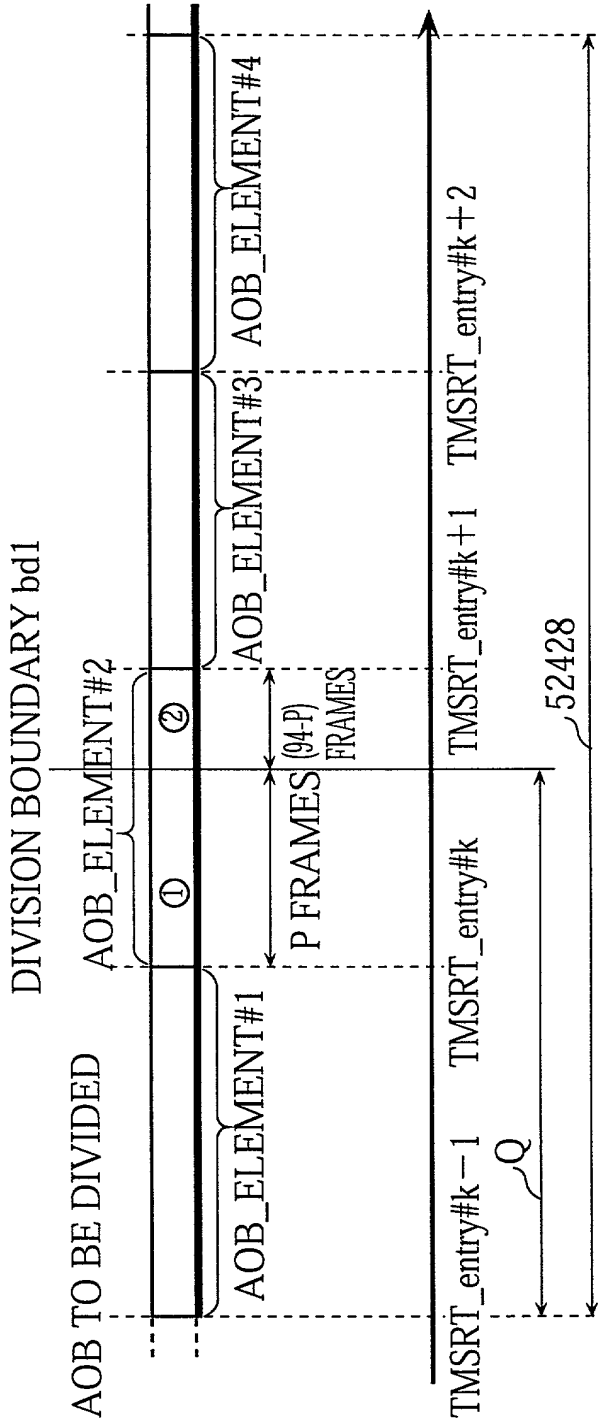


FIG. 35B

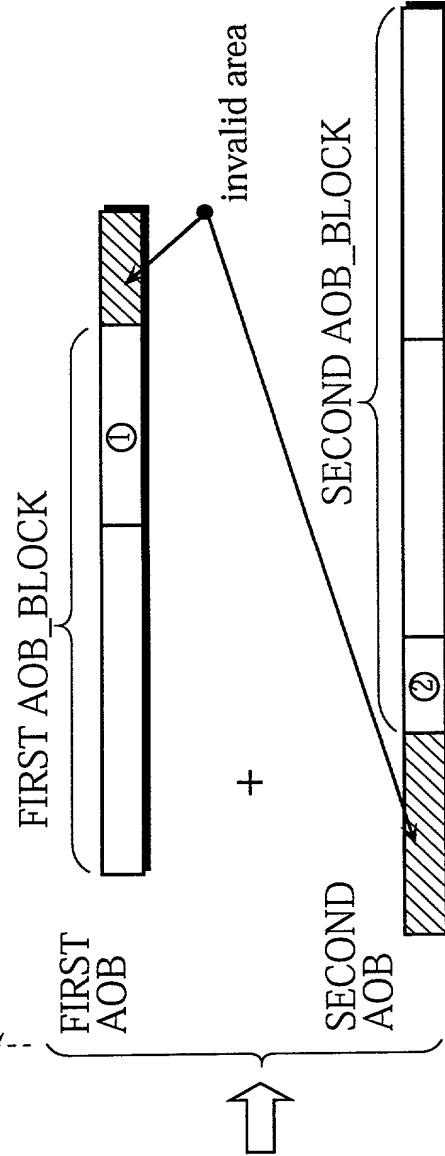






FIG. 37

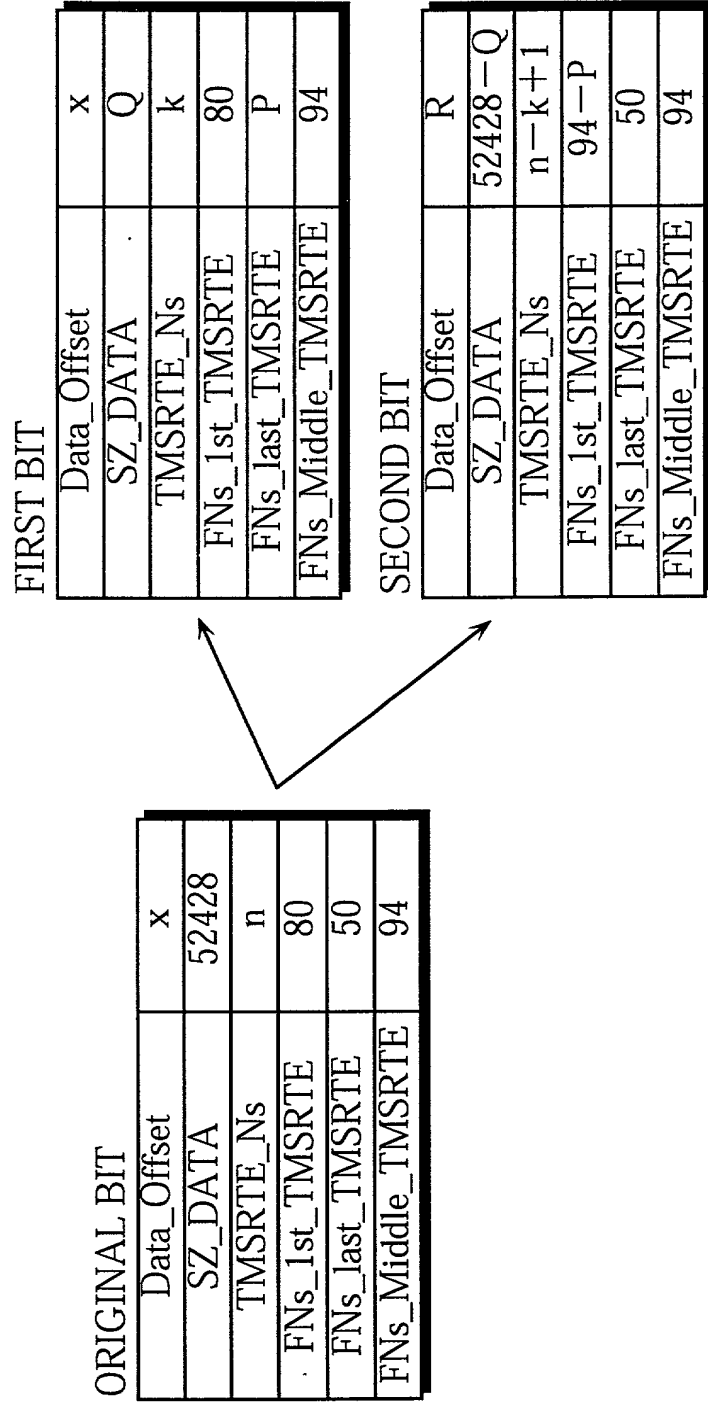




FIG. 39A

ATTRIBUTES OF DPL\_TK\_ATTR

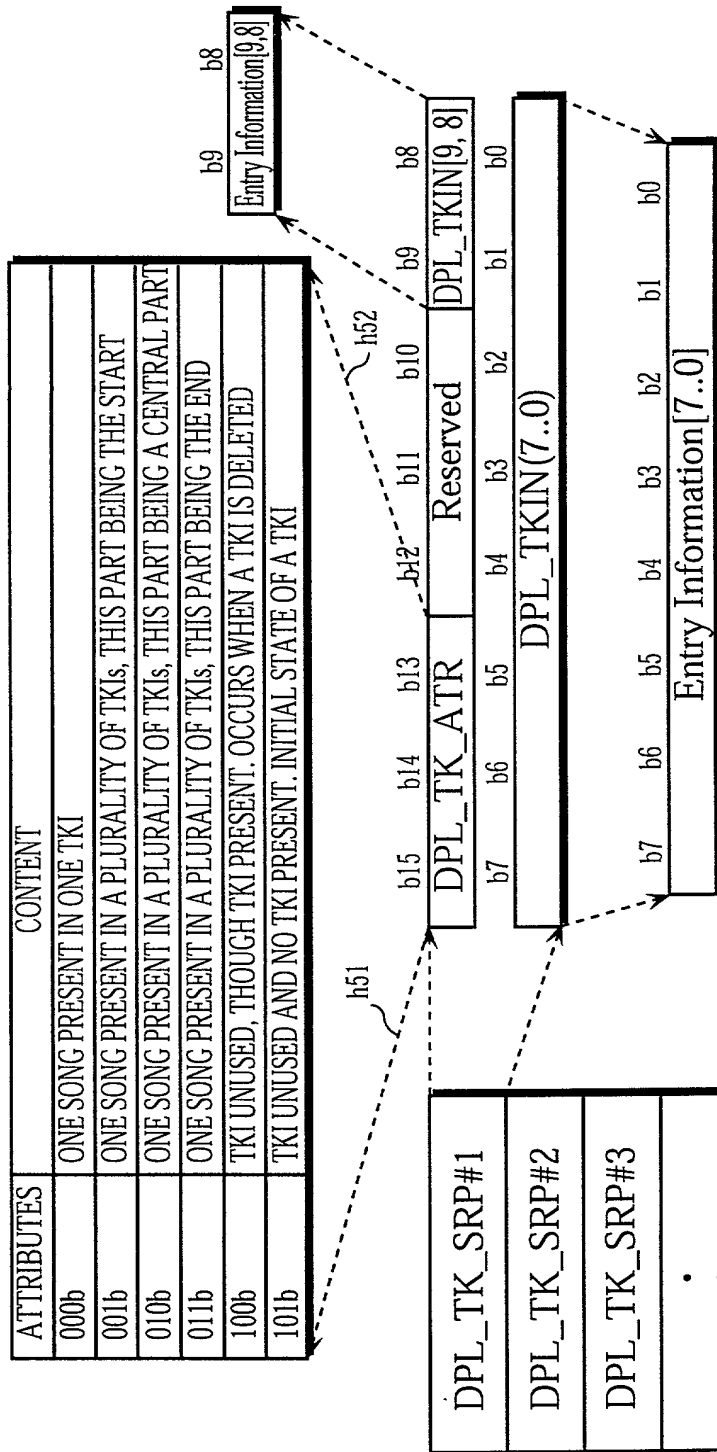


FIG. 39B



FIG. 40

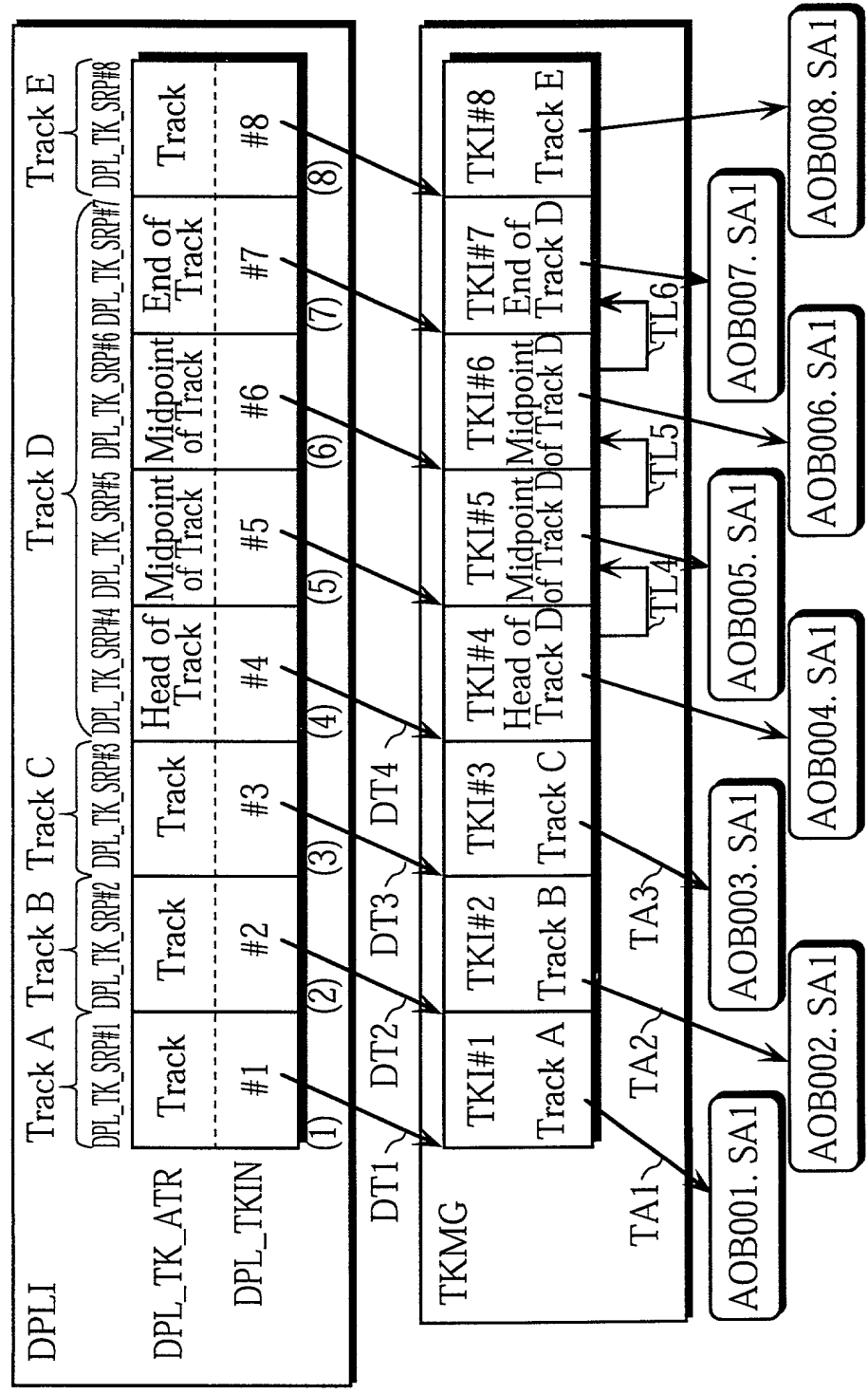


FIG. 41

DPLI	Track A			Track B			Track C			Track D			Track E											
	DPL_TK_SRP#1			DPL_TK_SRP#2			DPL_TK_SRP#3			DPL_TK_SRP#4			DPL_TK_SRP#5			DPL_TK_SRP#6			DPL_TK_SRP#7			DPL_TK_SRP#8		
	Track			Track			Track			Head of Track			Midpoint of Track			Midpoint of Track			End of Track			Track		
DPL_TK_ATR	#1			#2			#3			#4			#5			#6			#7			#8		
DPL_TKIN																								

FIG. 42

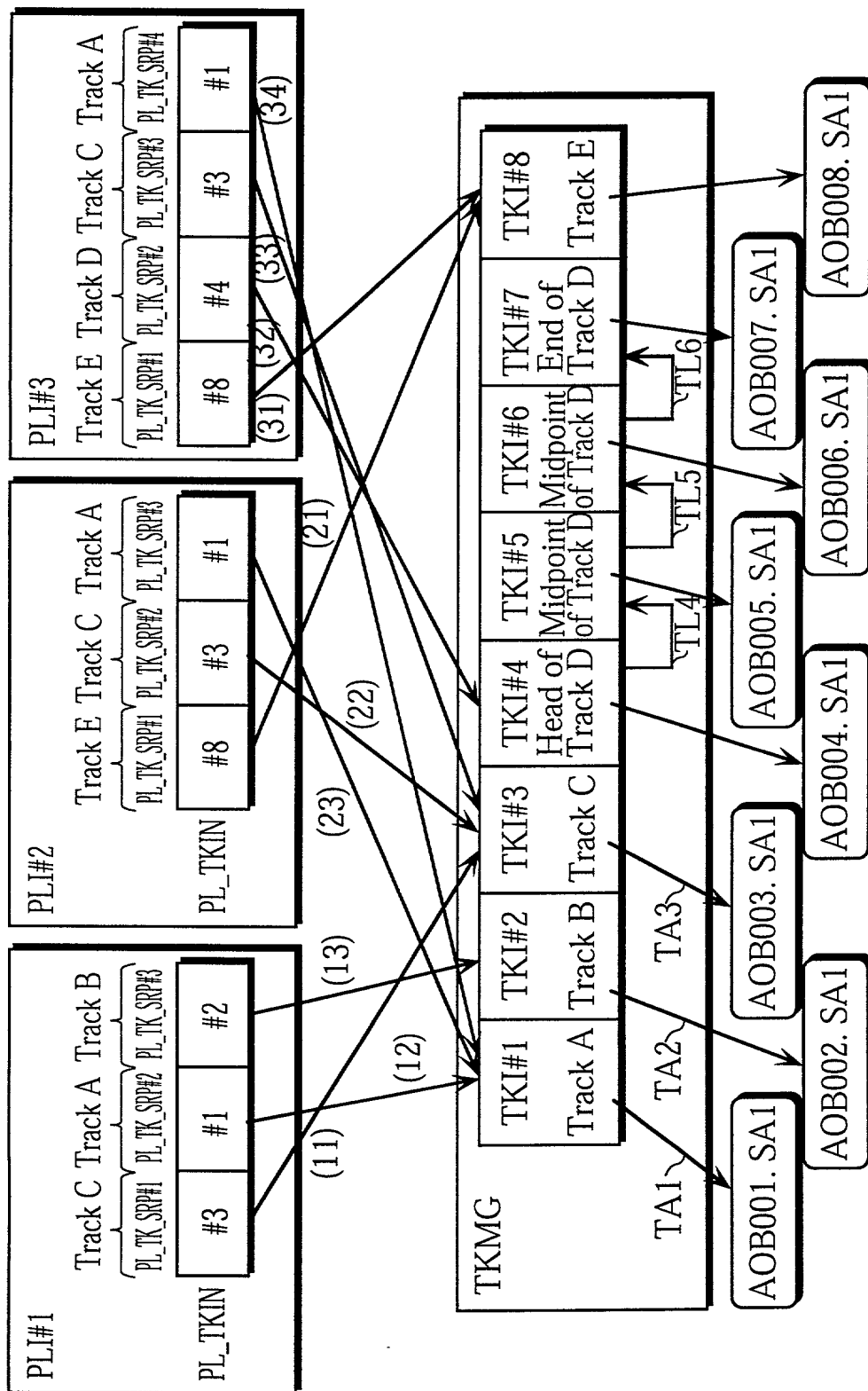


FIG. 43A

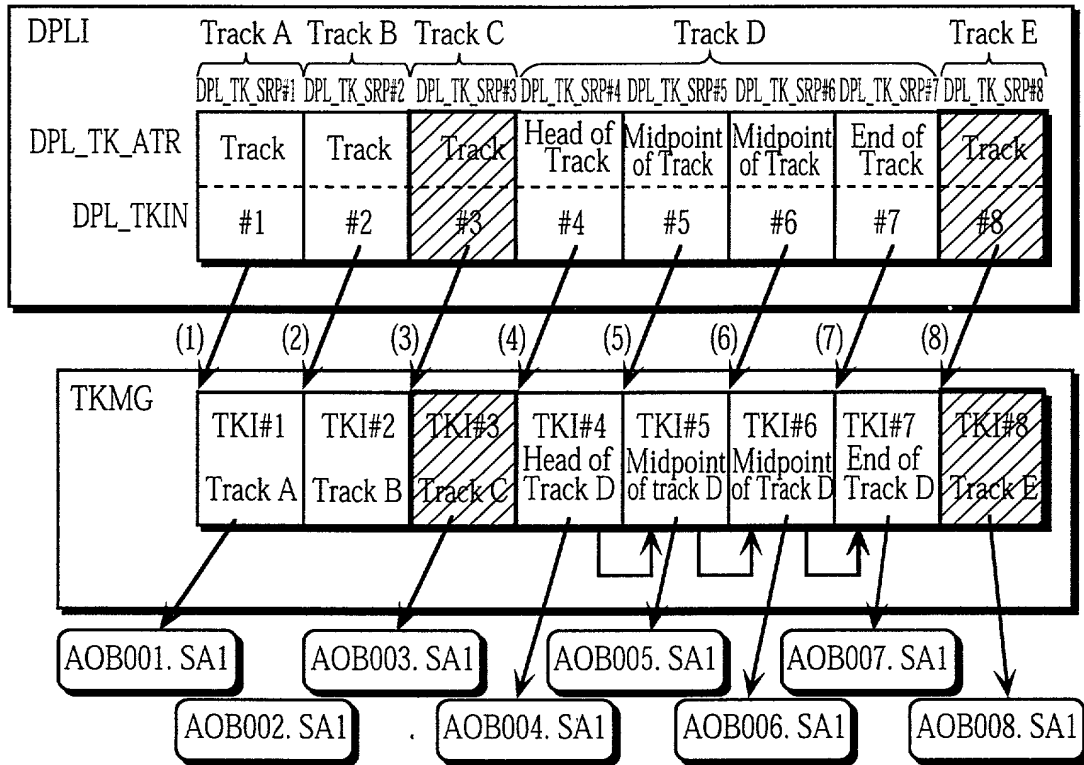


FIG. 43B

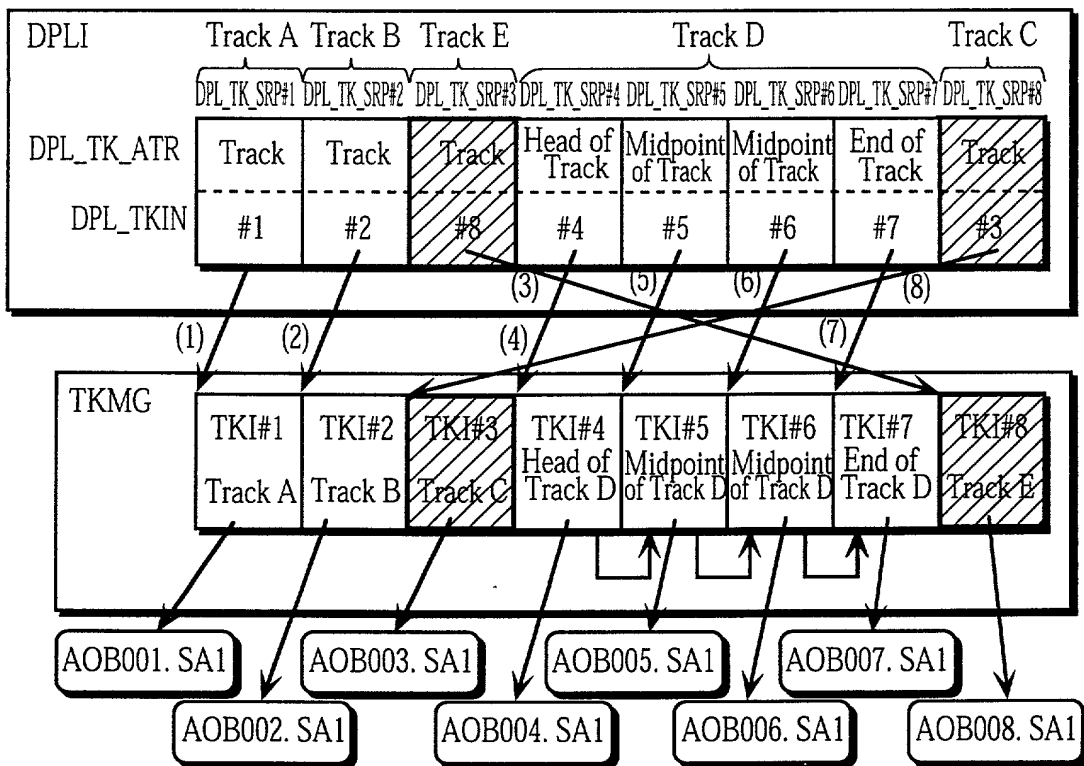




FIG. 44A

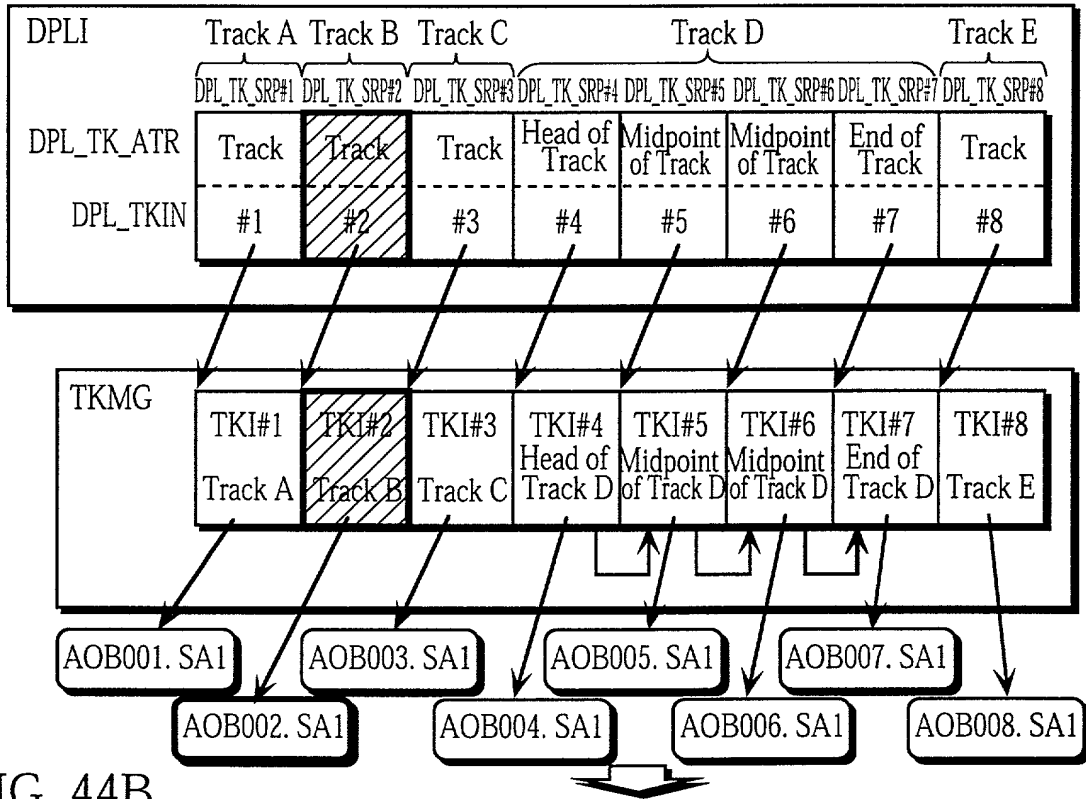


FIG. 44B

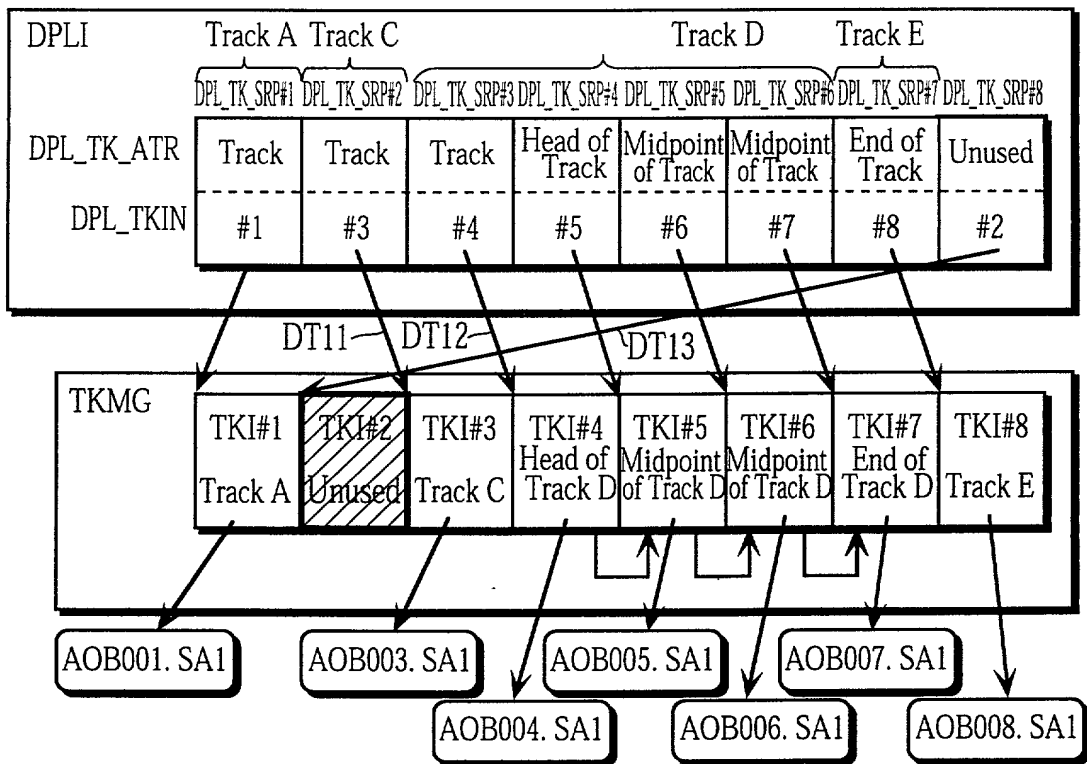


FIG. 45A

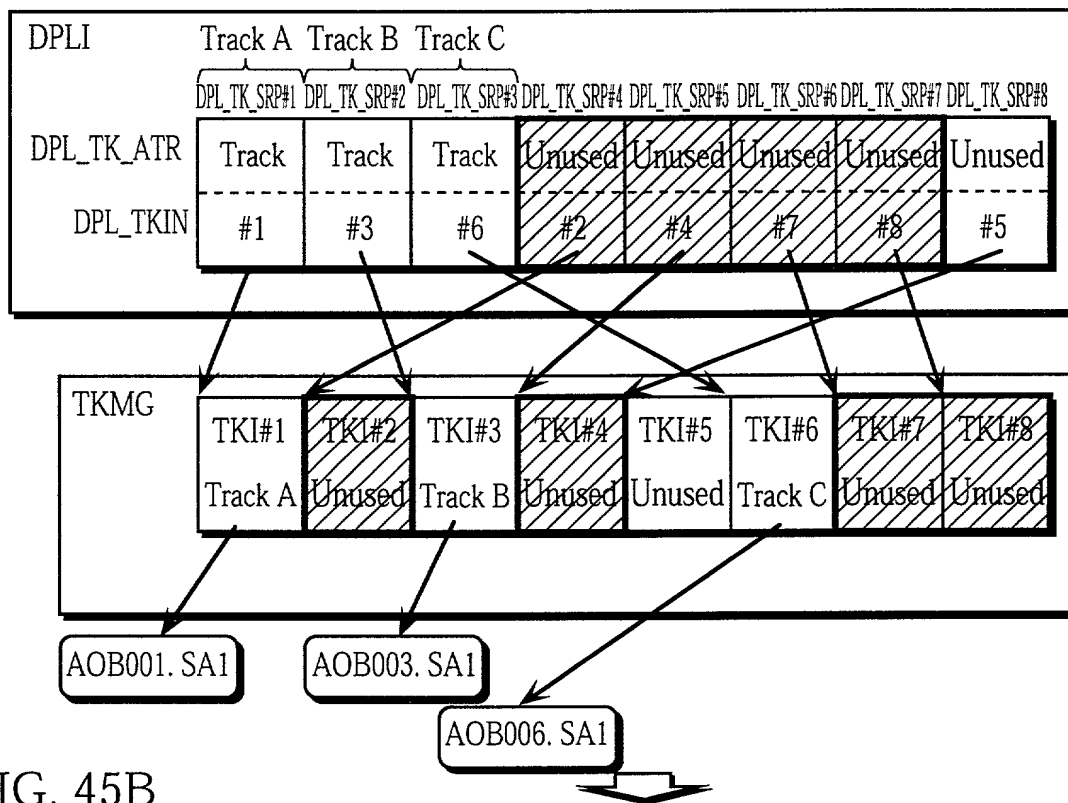


FIG. 45B

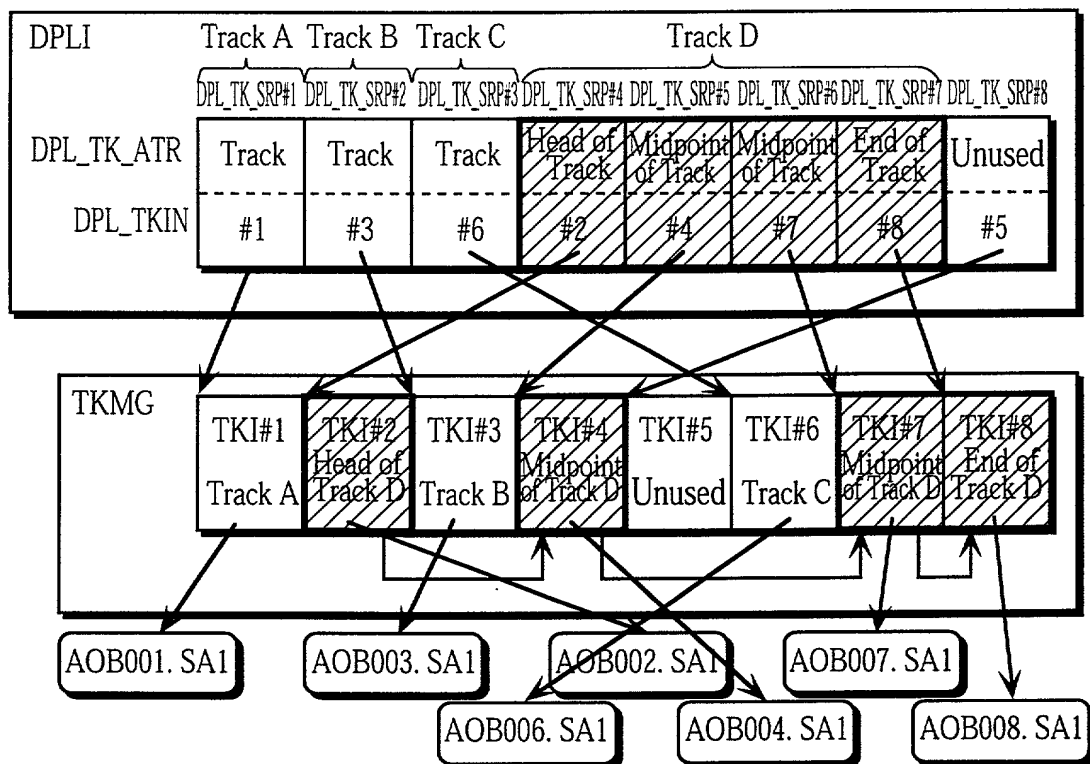


FIG. 46A

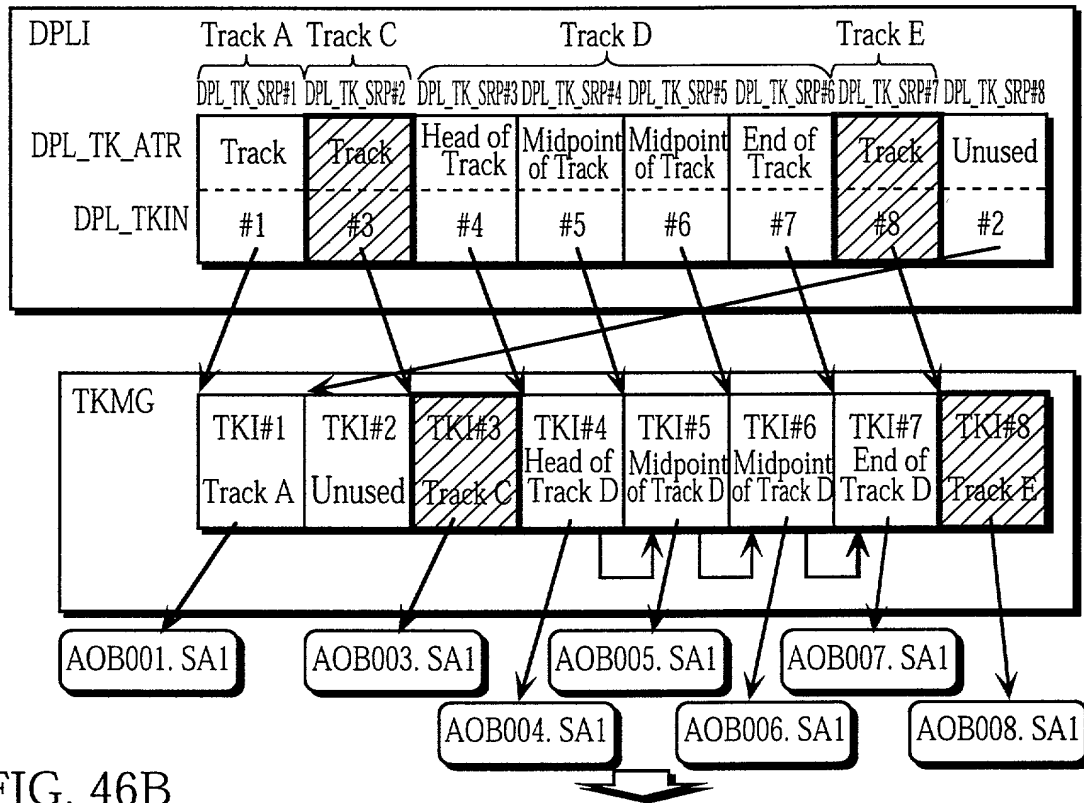


FIG. 46B

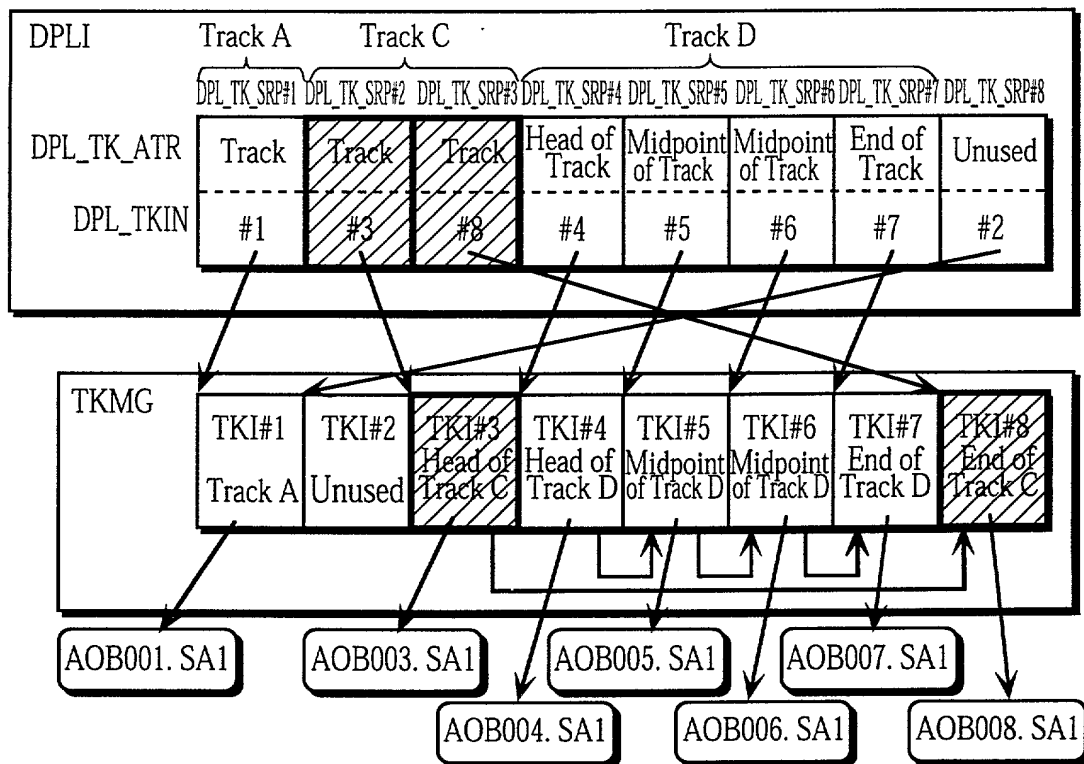


FIG. 47A

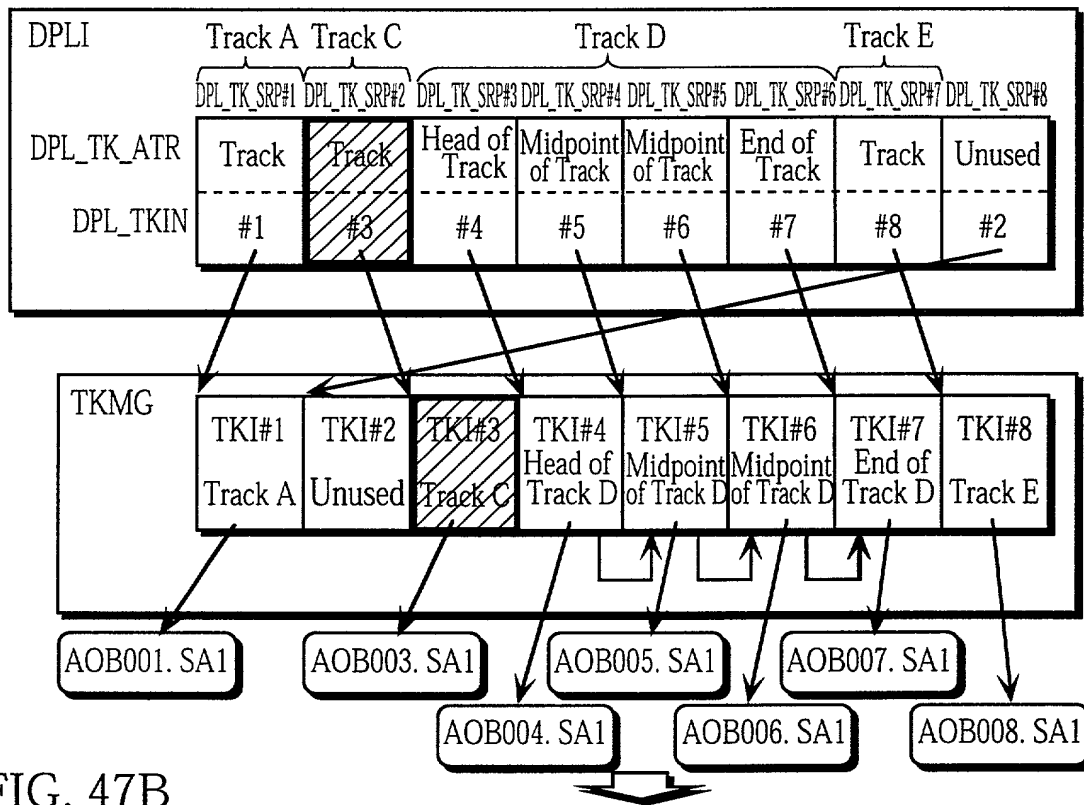


FIG. 47B

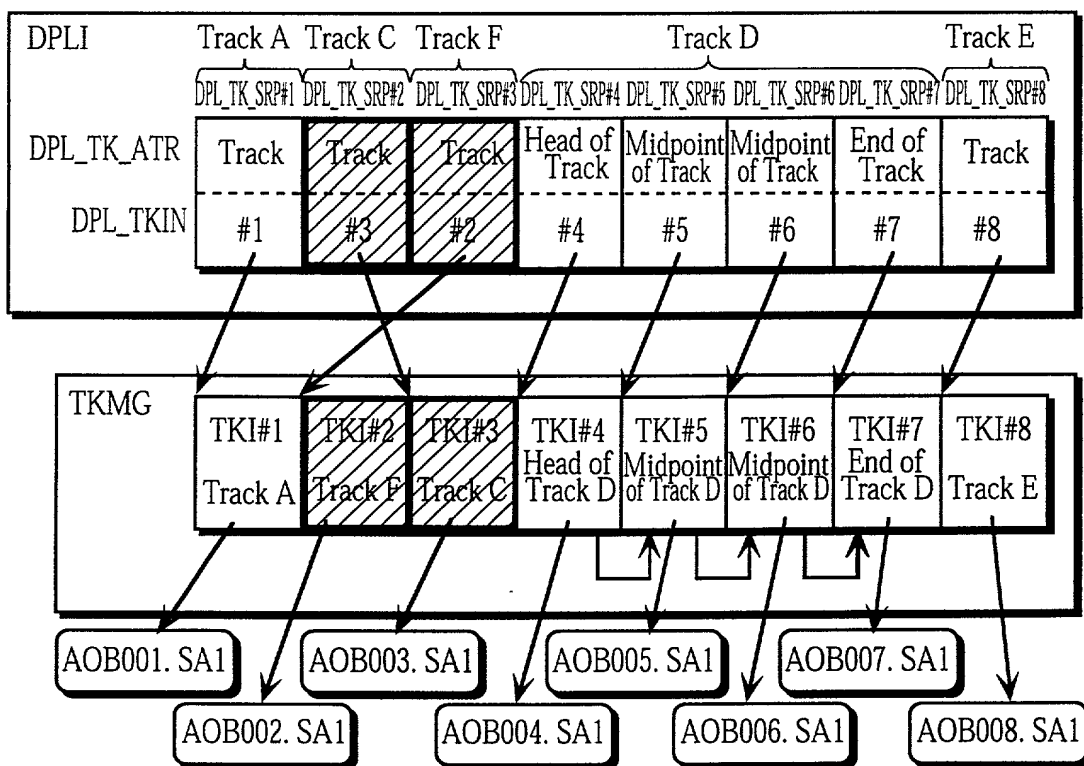




FIG. 49

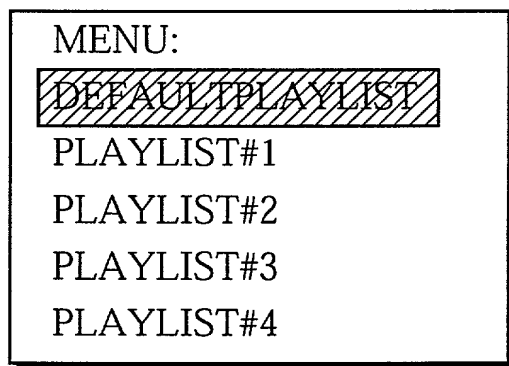


FIG. 50A

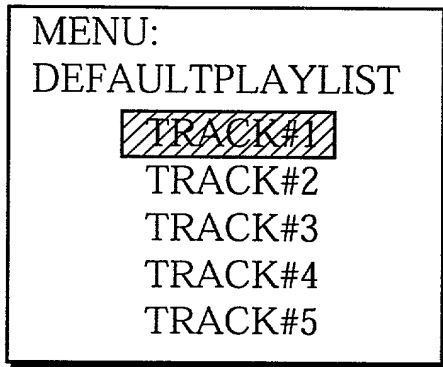


FIG. 50B

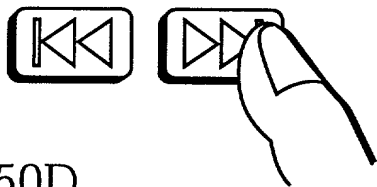
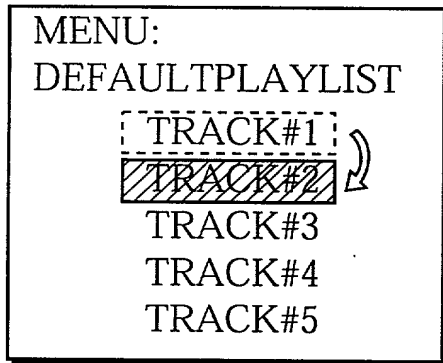


FIG. 50C

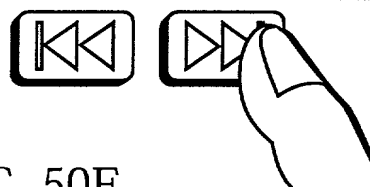
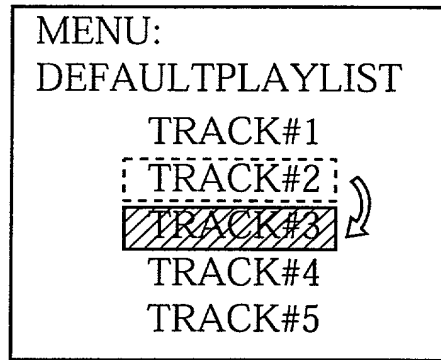


FIG. 50D

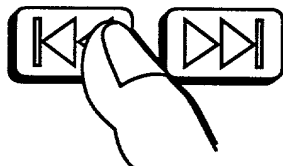
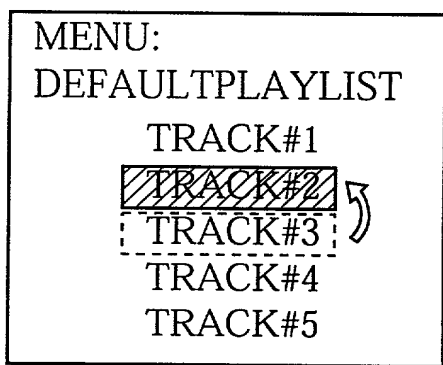
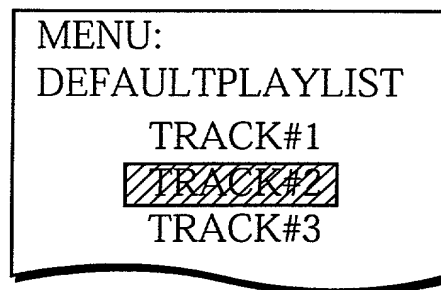


FIG. 50E



PLAY ...INDICATES THAT TRACK#2 SHOULD BE PLAYED BACK  
Playlist ...INDICATES THAT TRACK#2 SHOULD BE EDITED

FIG. 51A

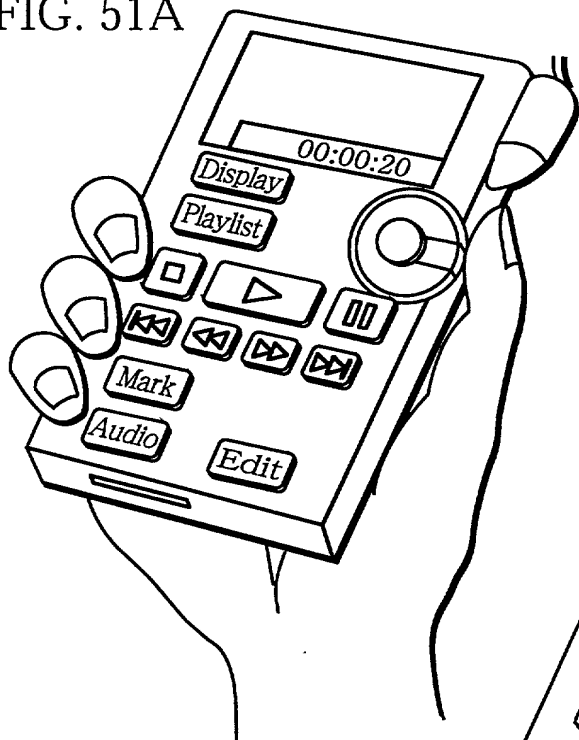


FIG. 51C

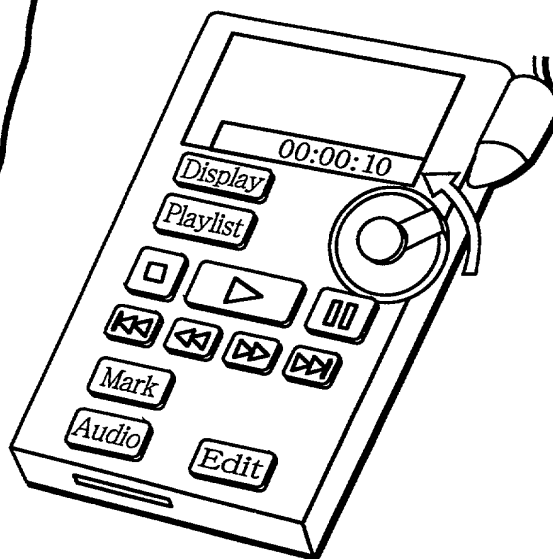
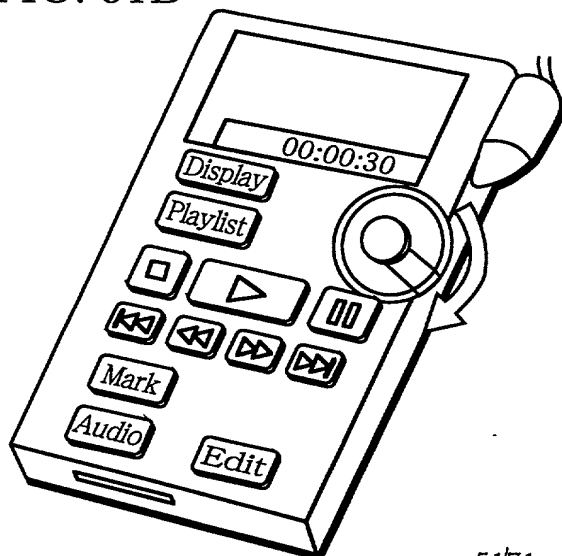


FIG. 51B





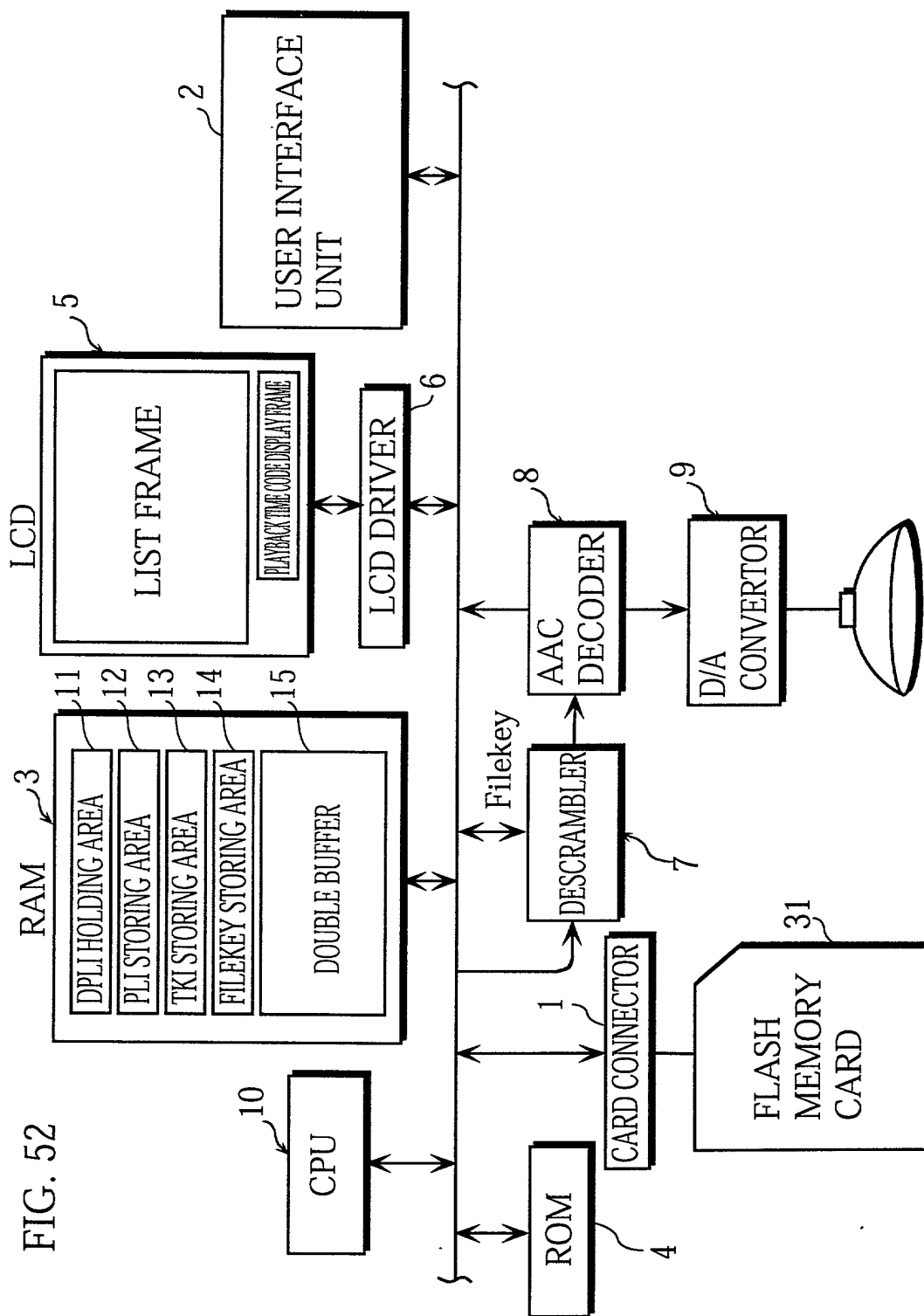


FIG. 53

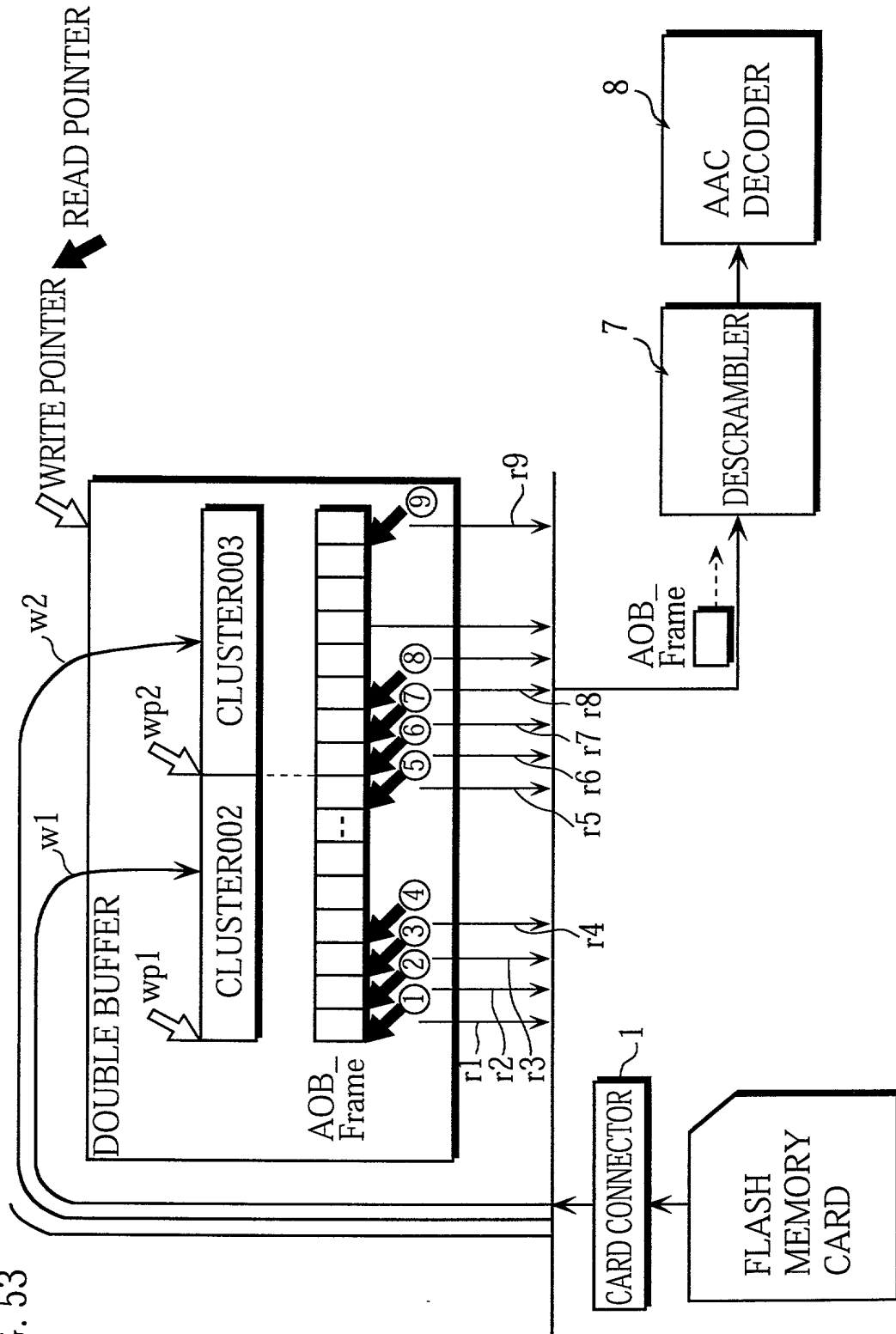


FIG. 54A

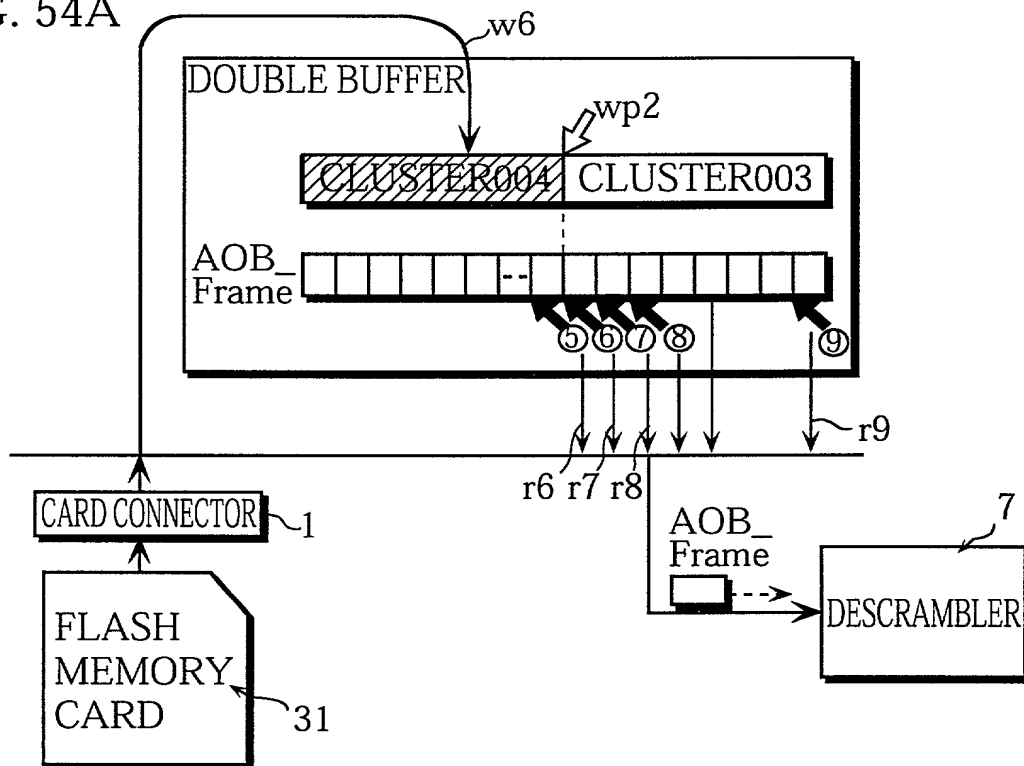


FIG. 54B

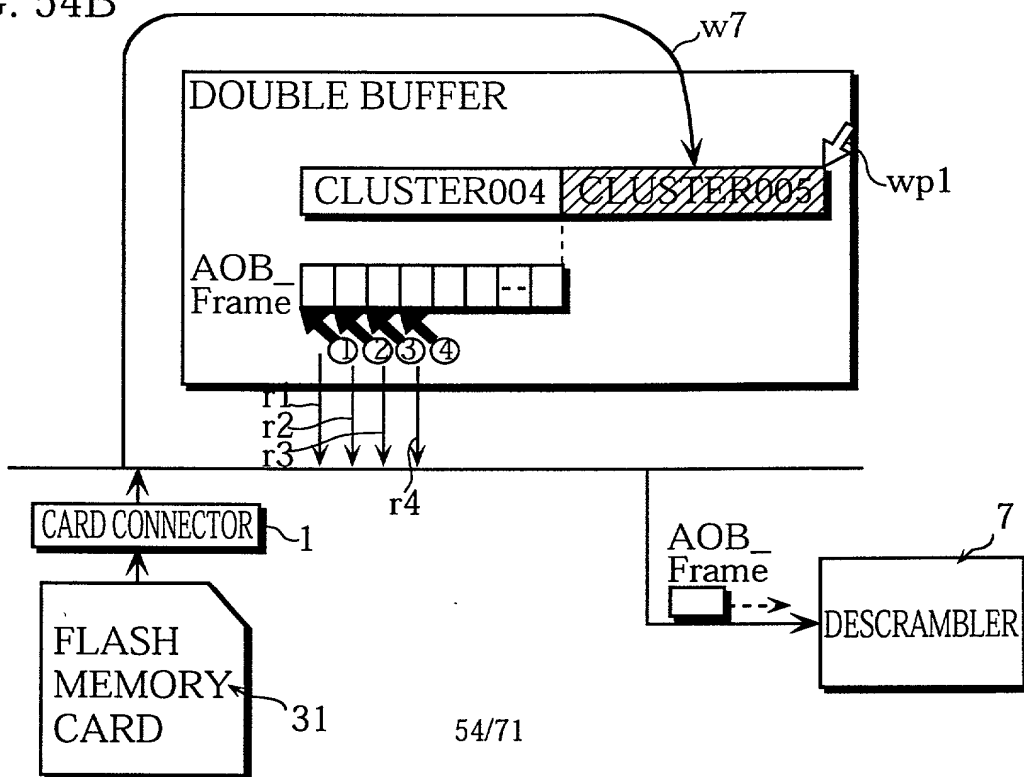


FIG. 55

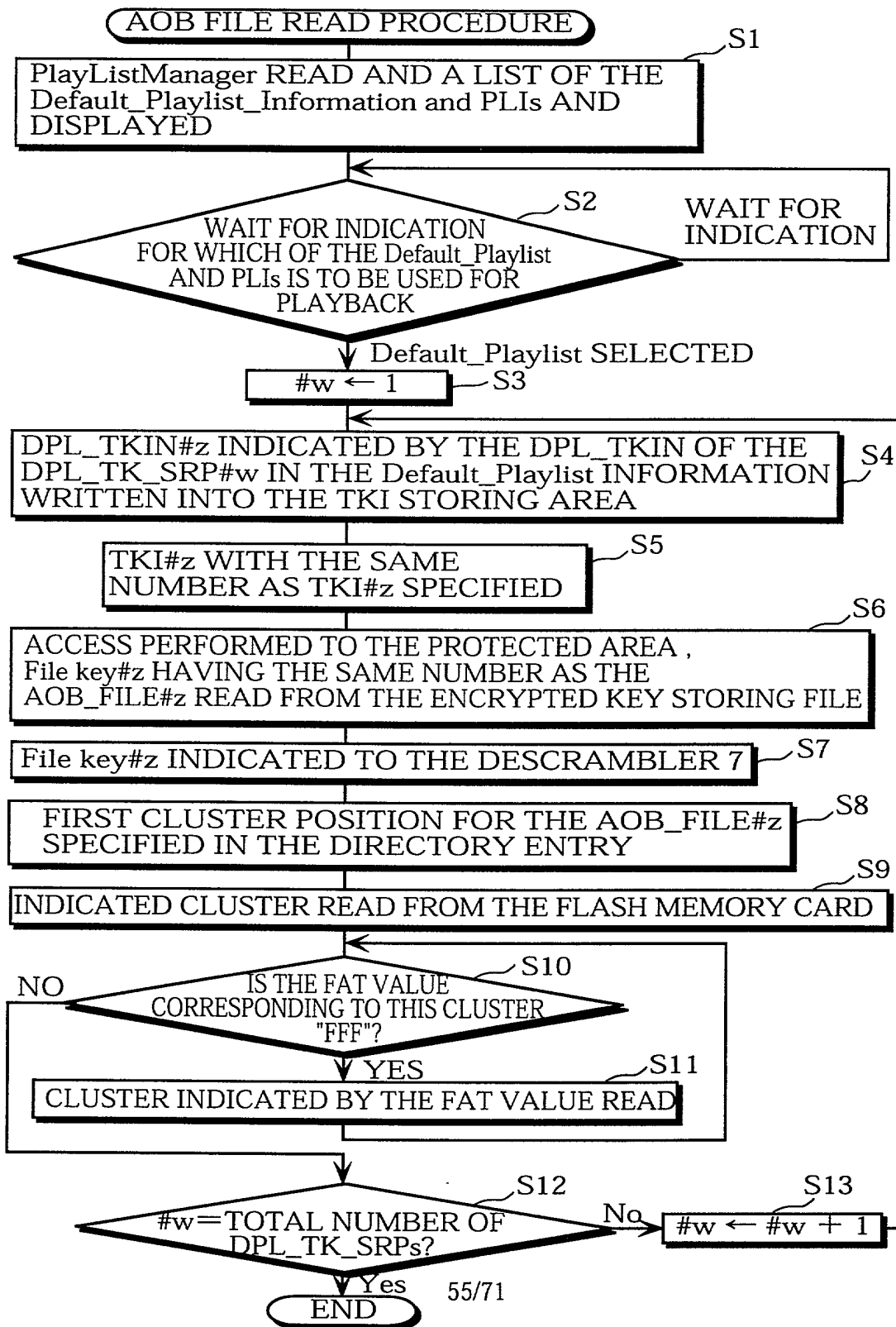


FIG. 56

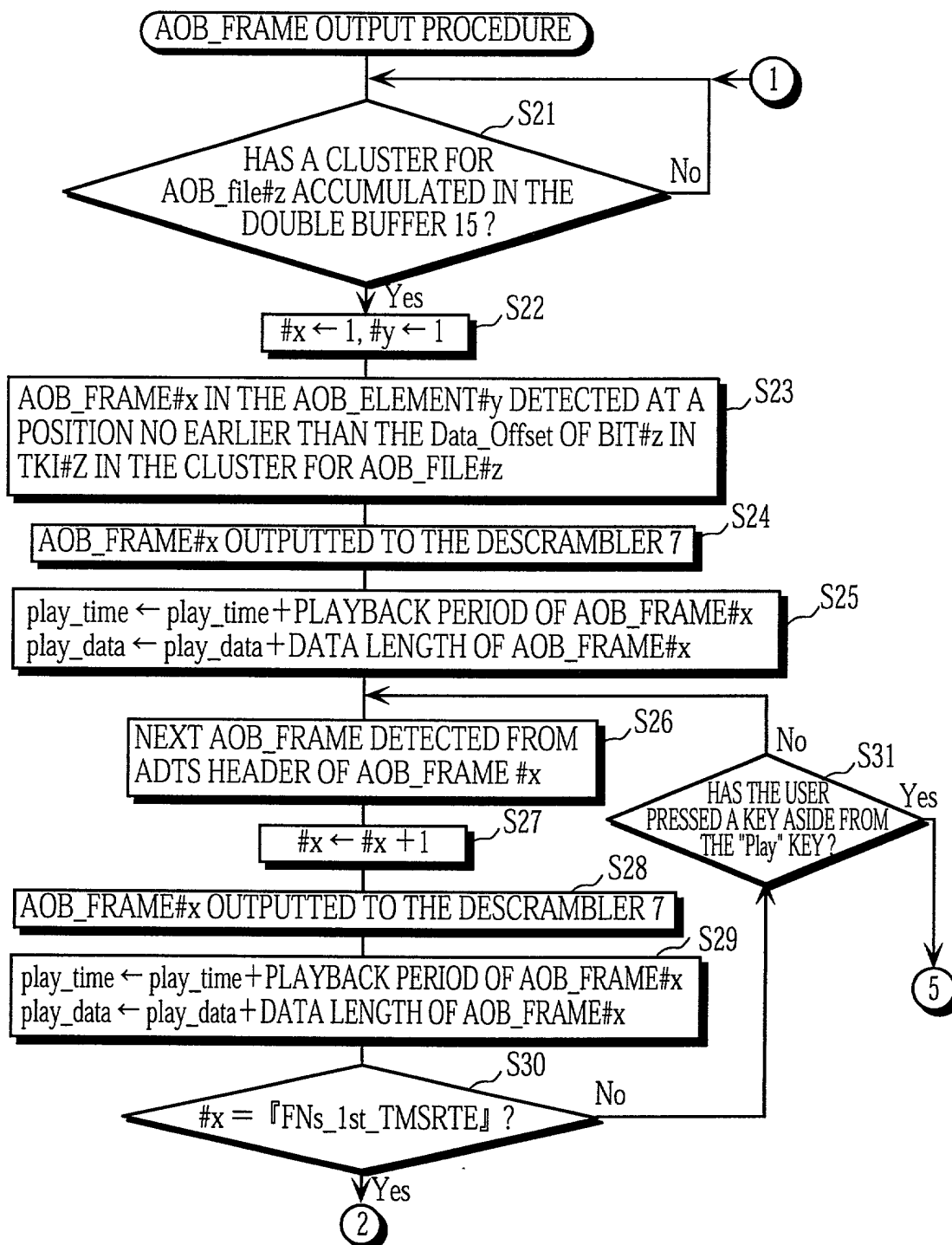


FIG. 57

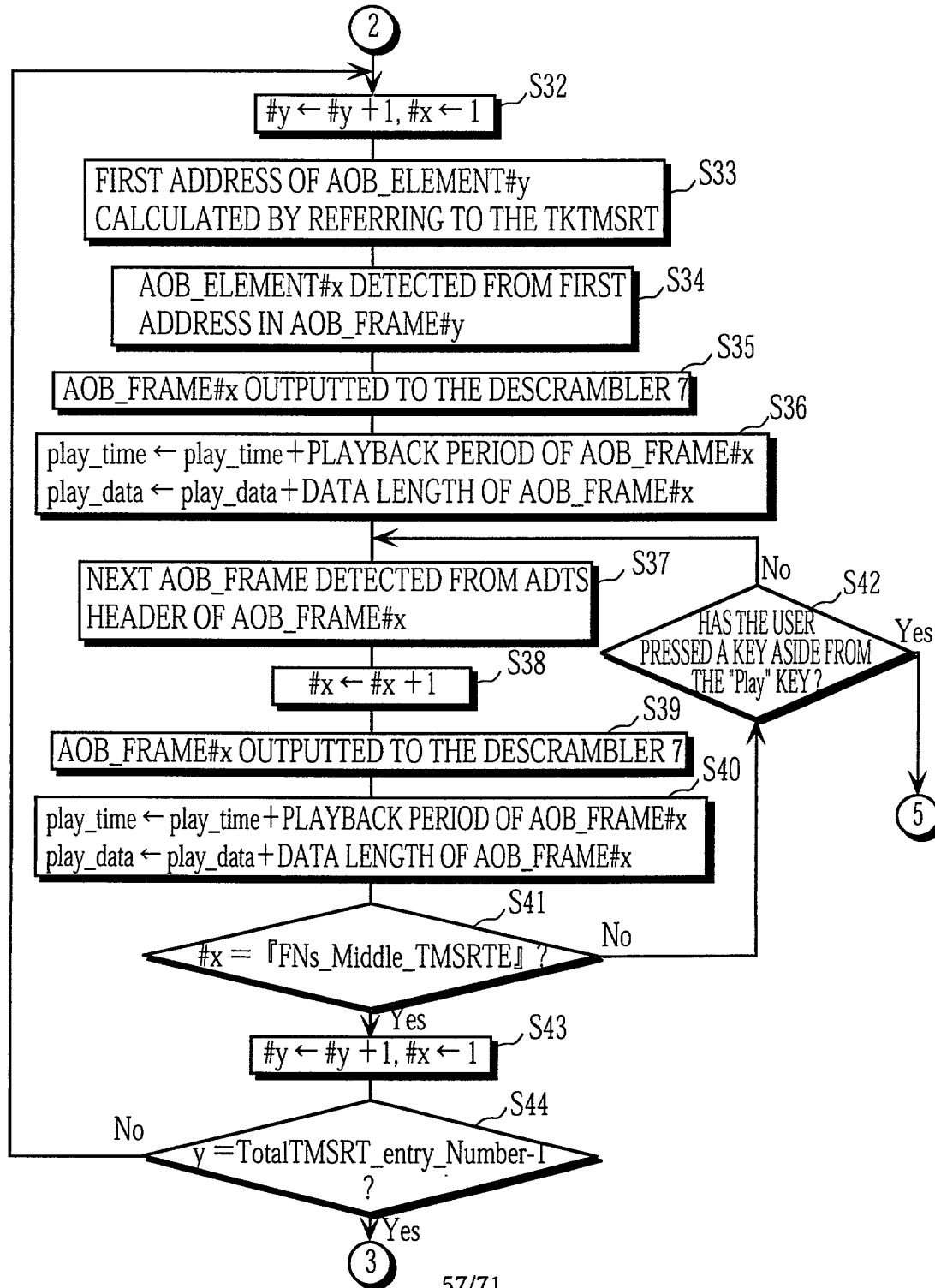


FIG. 58

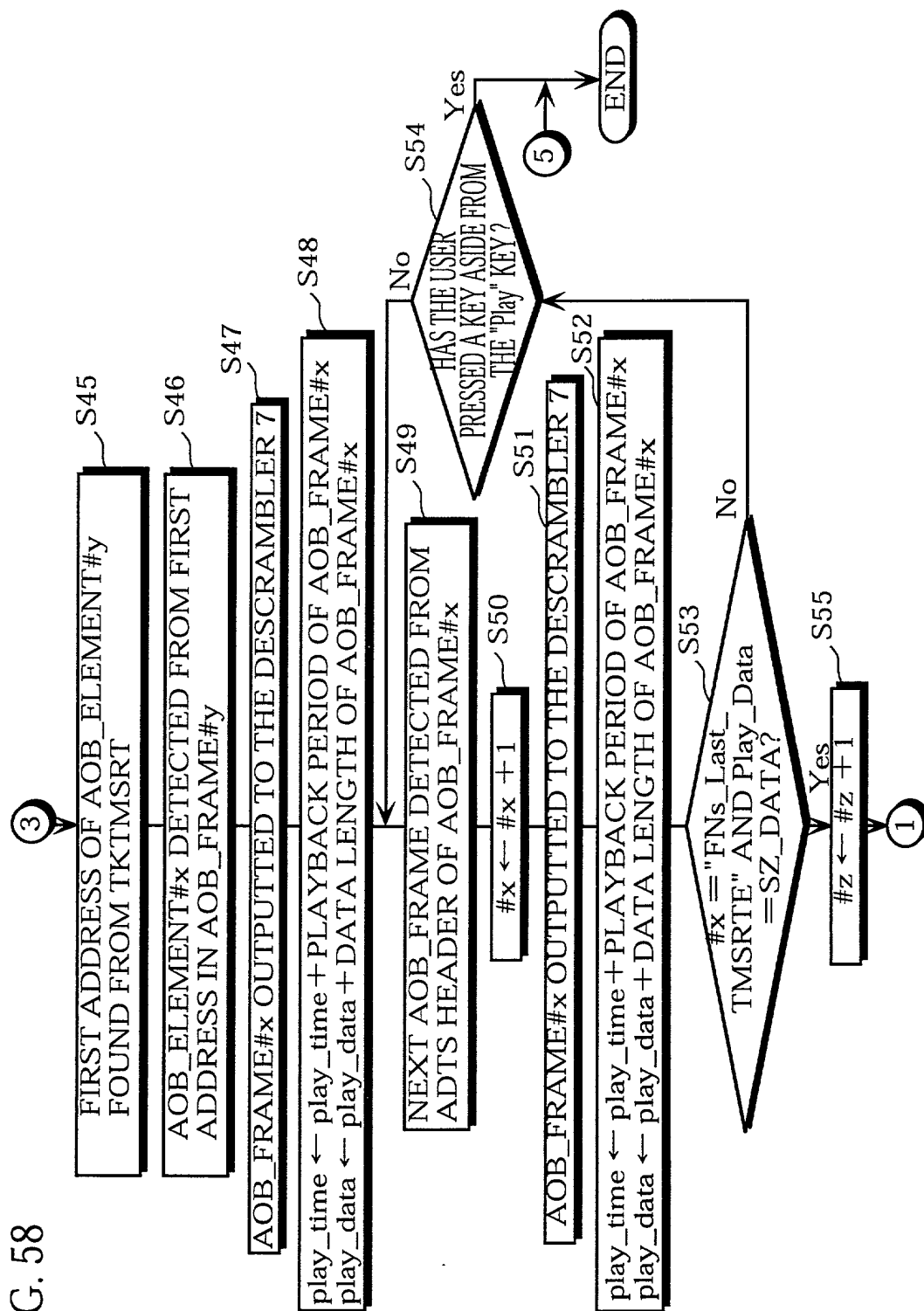
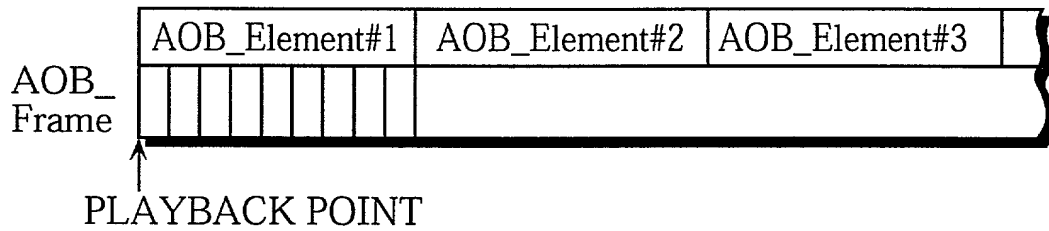
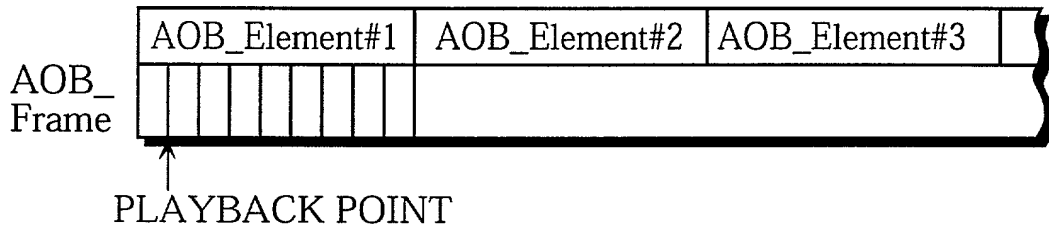


FIG. 59A



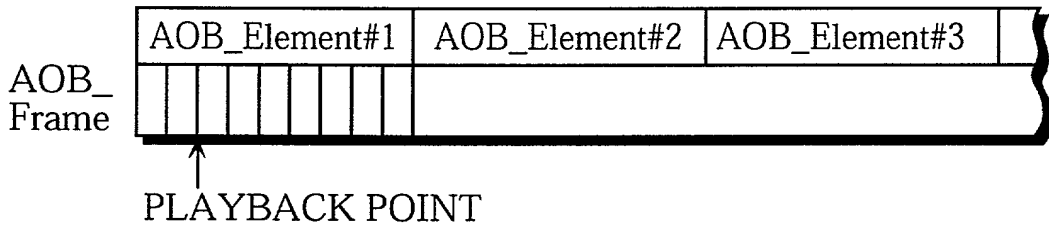
PLAYBACK TIME CODE=00:00:00.000

FIG. 59B



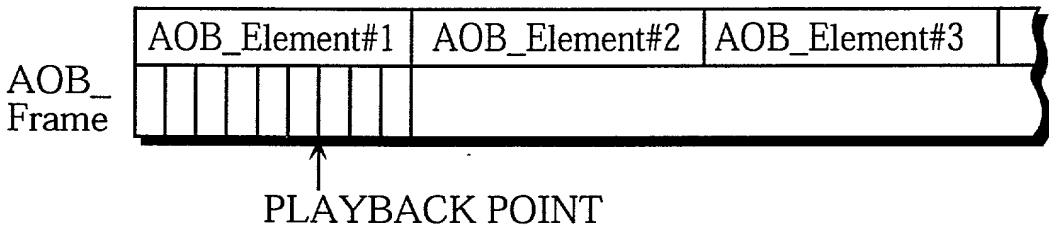
PLAYBACK TIME CODE=00:00:00.020

FIG. 59C



PLAYBACK TIME CODE=00:00:00.040

FIG. 59D



PLAYBACK TIME CODE=00:00:00.120



FIG. 60

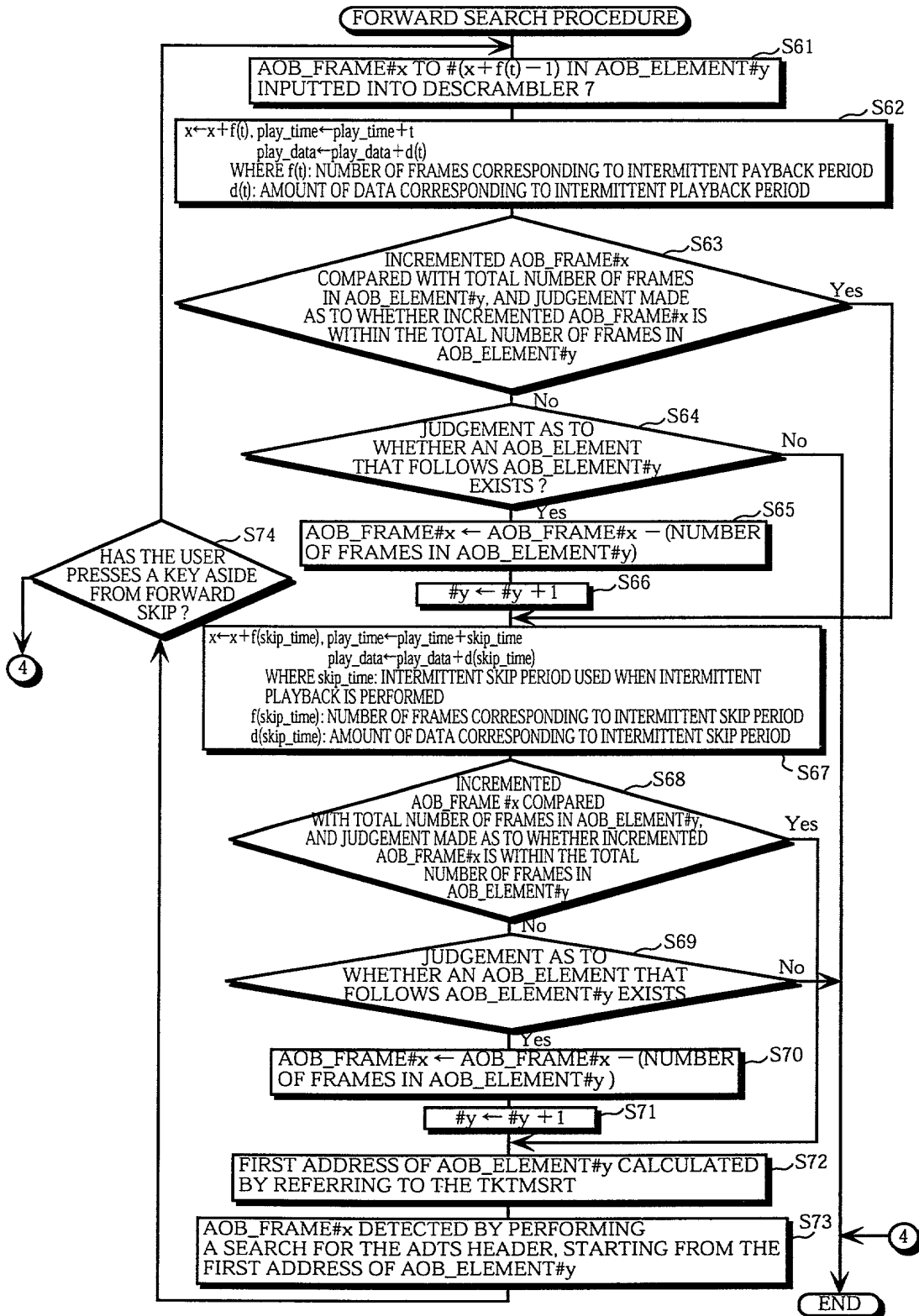


FIG. 61A

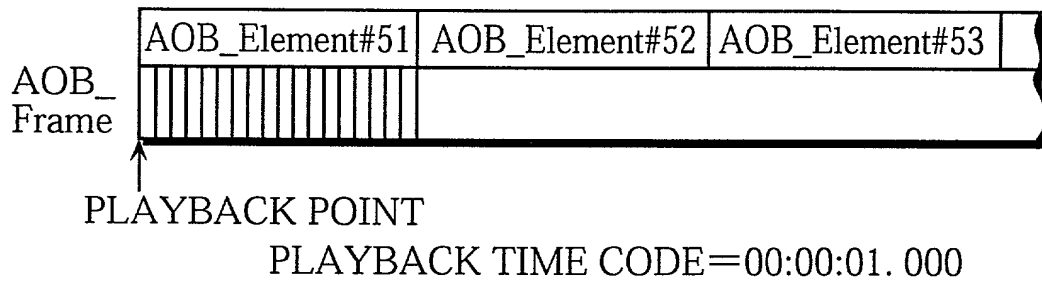


FIG. 61B

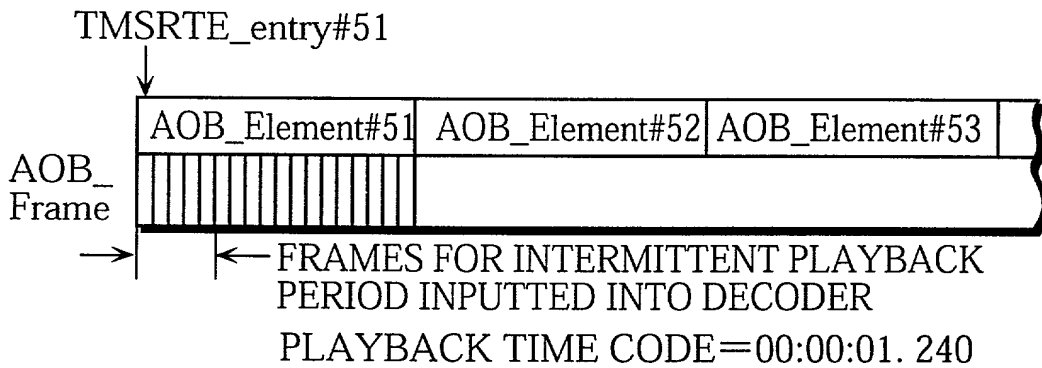
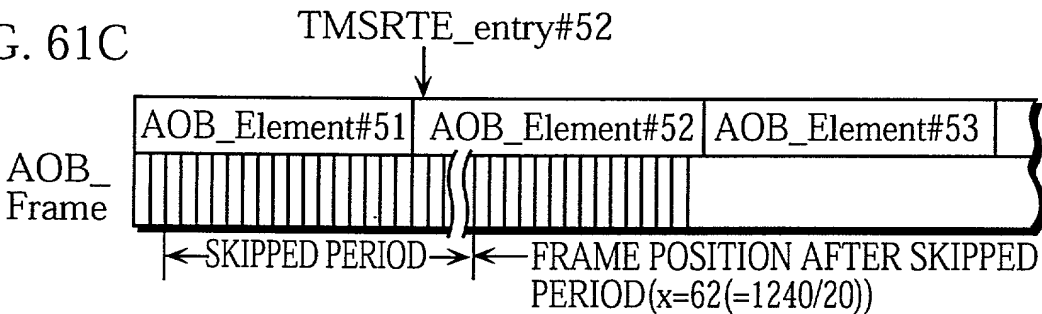
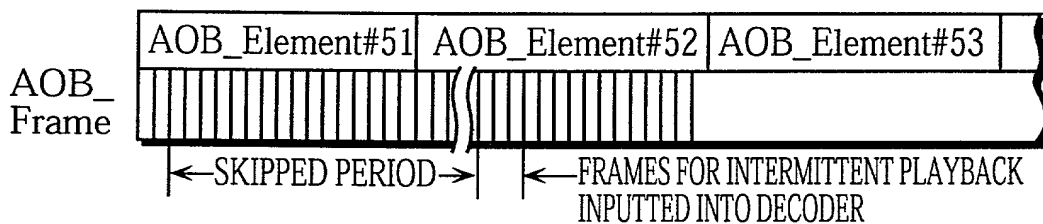


FIG. 61C



PLAYBACK TIME CODE=00:00:03.240(2.000sec+1.240sec)

FIG. 61D



PLAYBACK TIME CODE=00:00:03.480

FIG. 62A

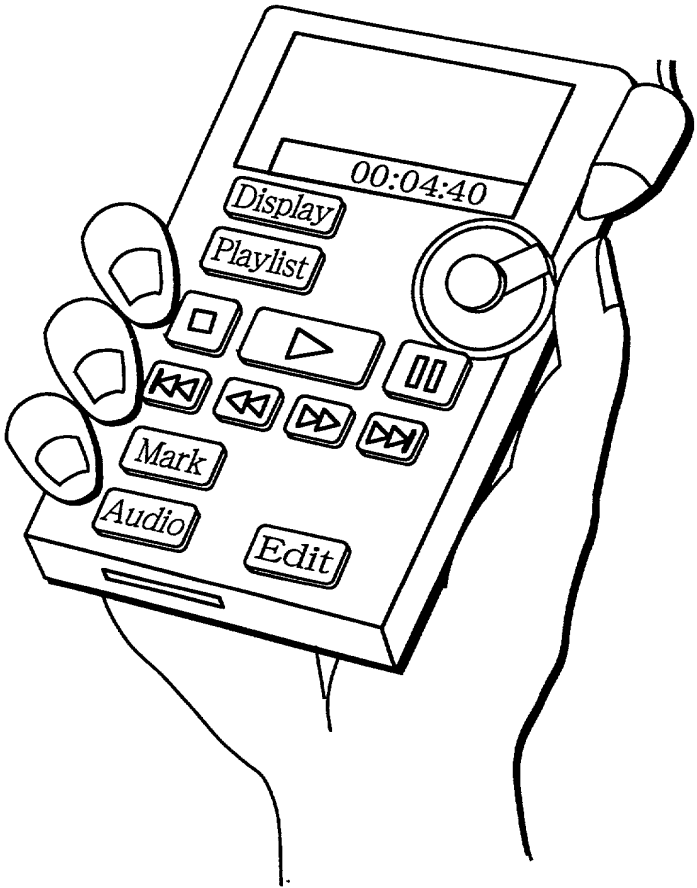


FIG. 62B

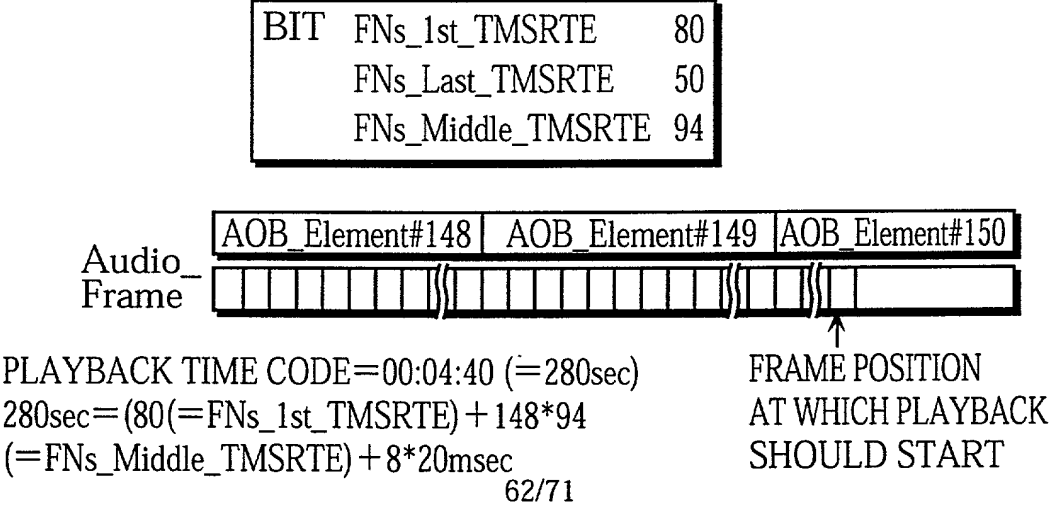


FIG. 63

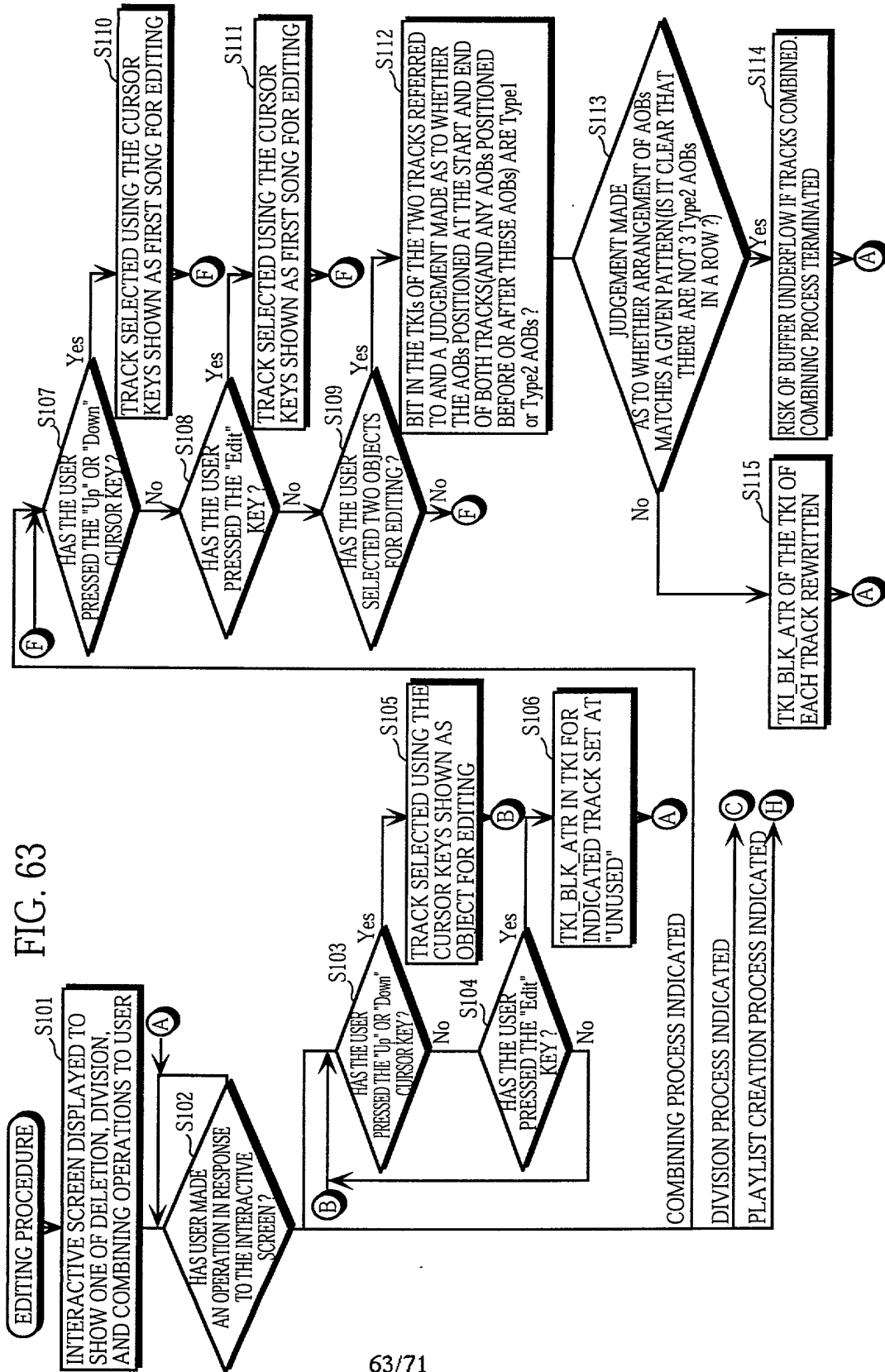


FIG. 64

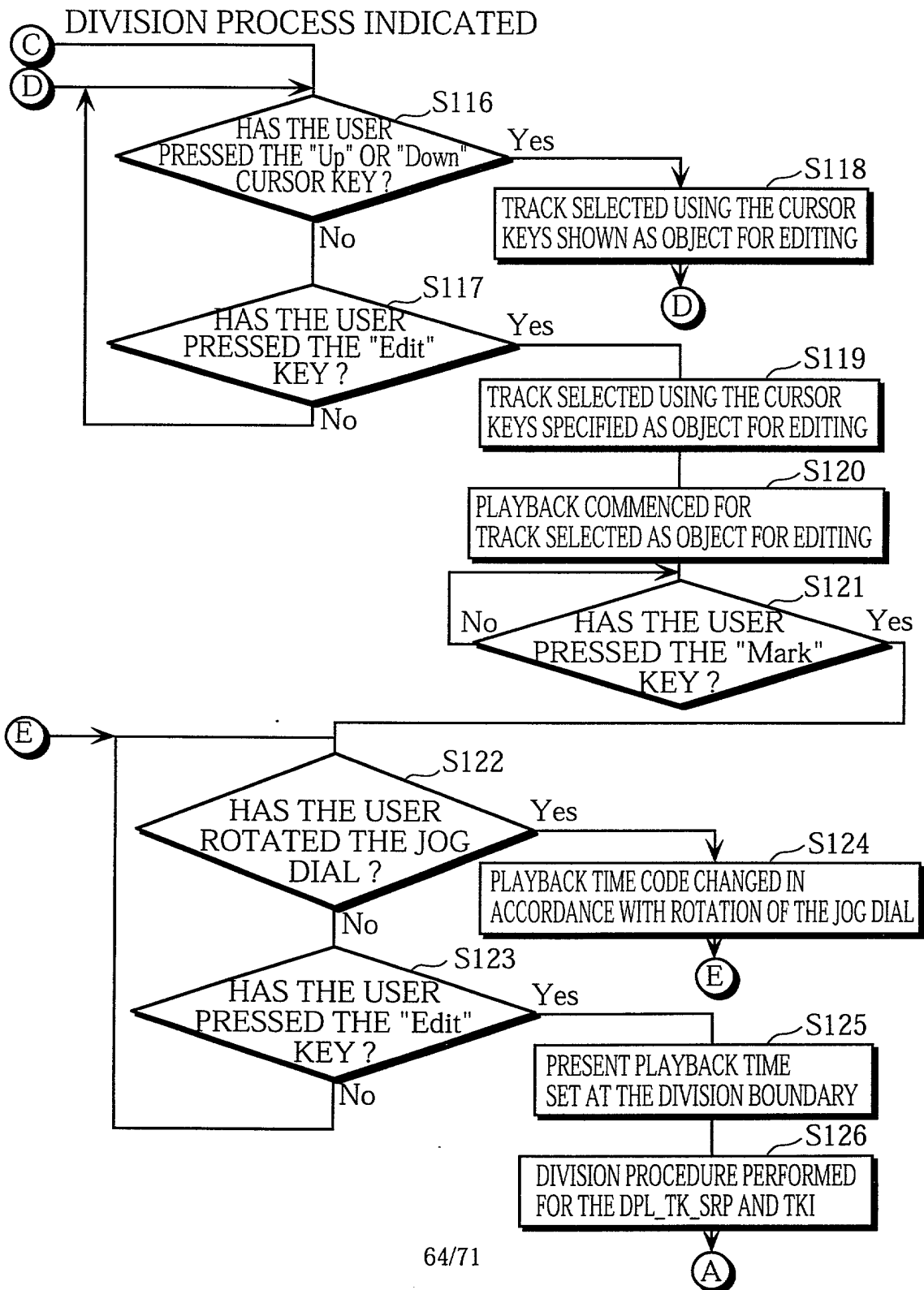


FIG. 65

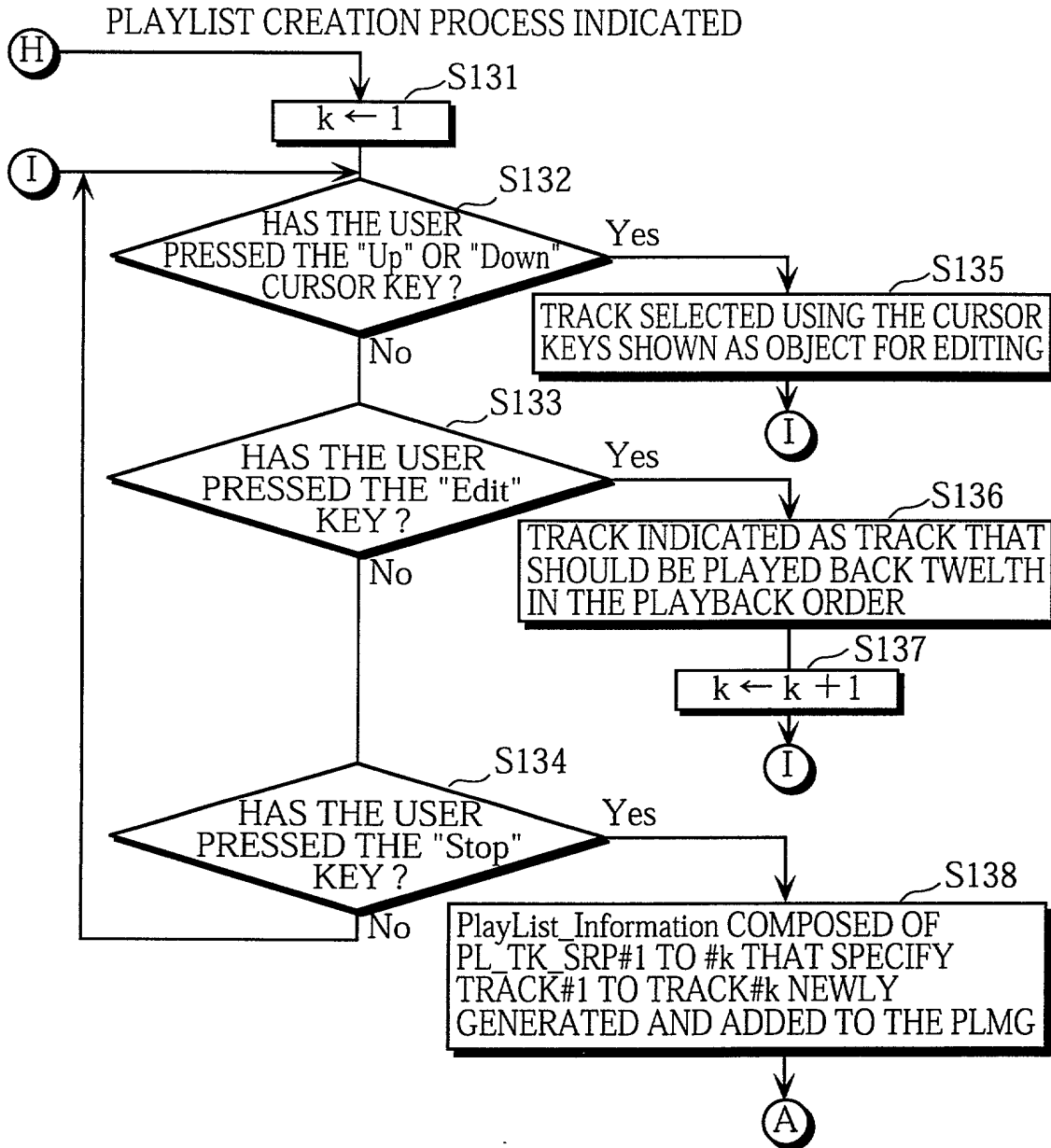
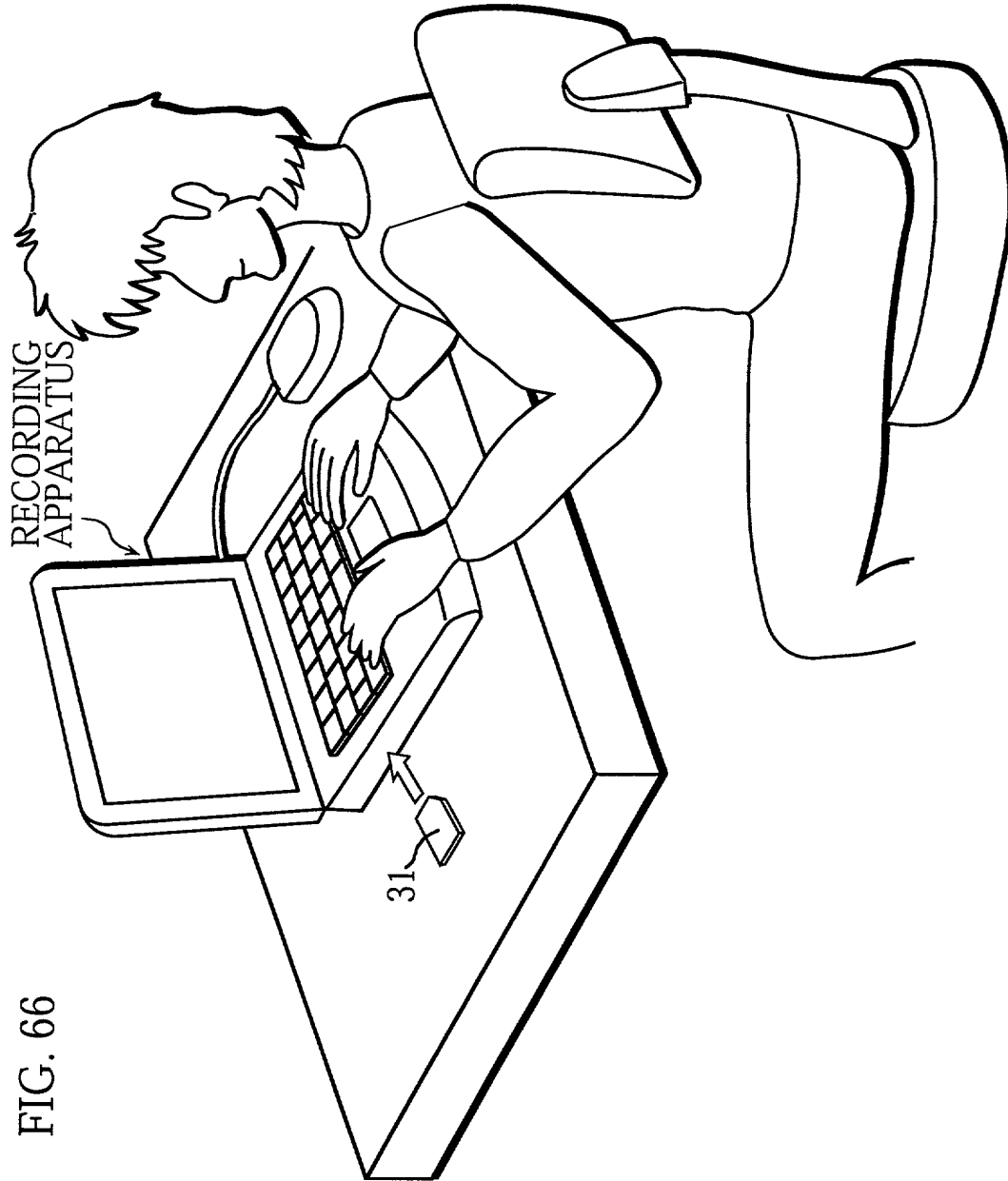


FIG. 66



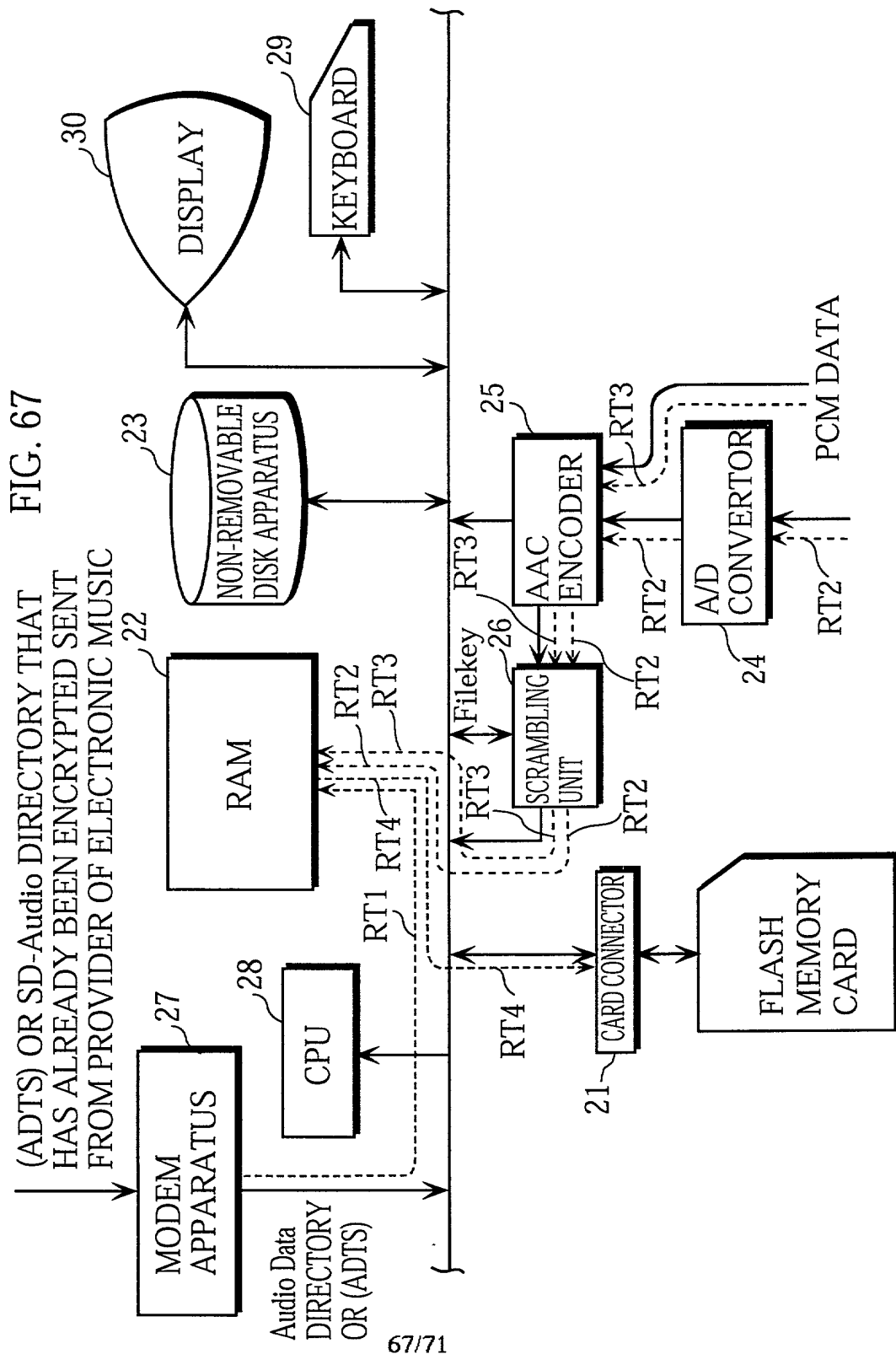




FIG. 68

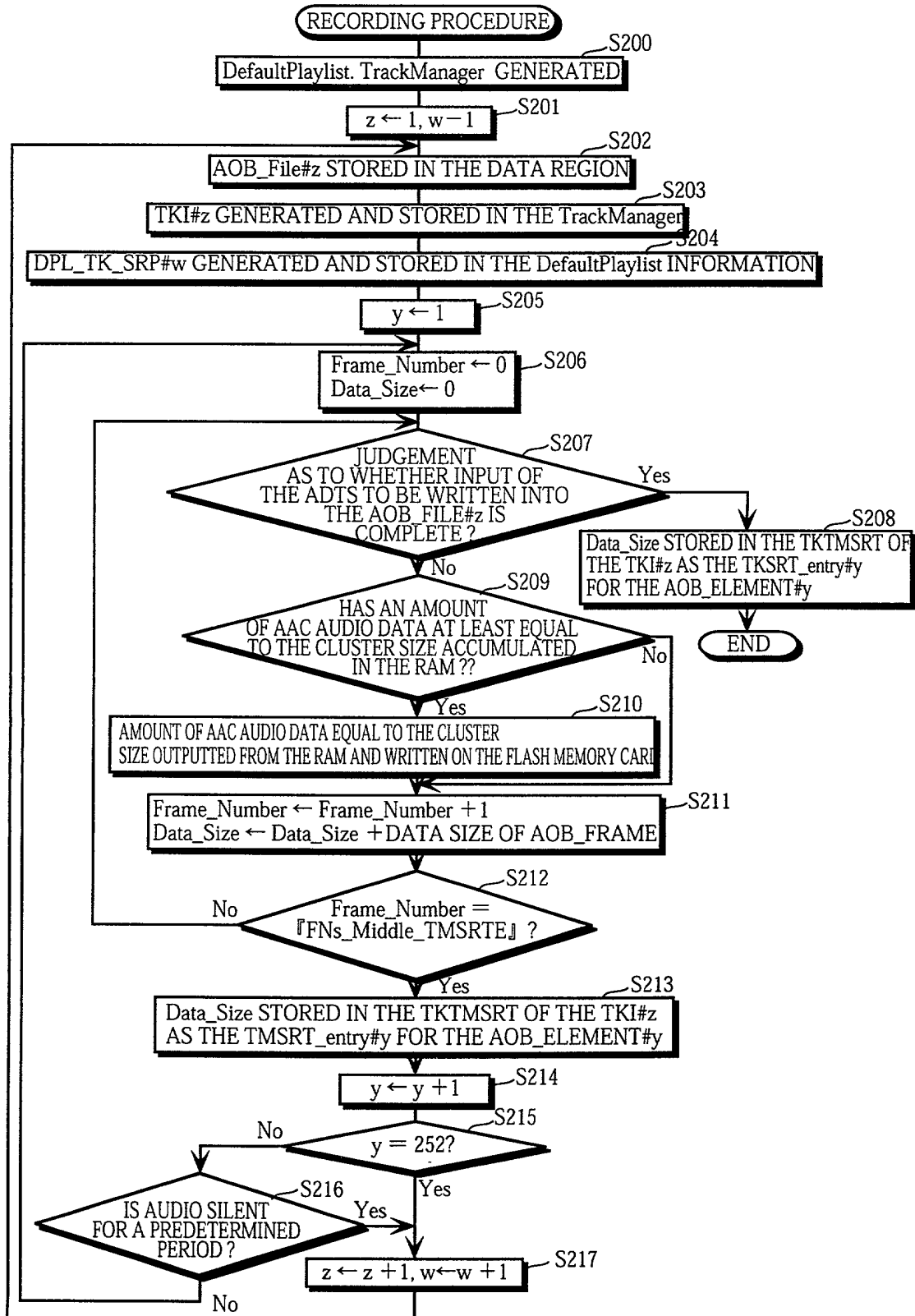
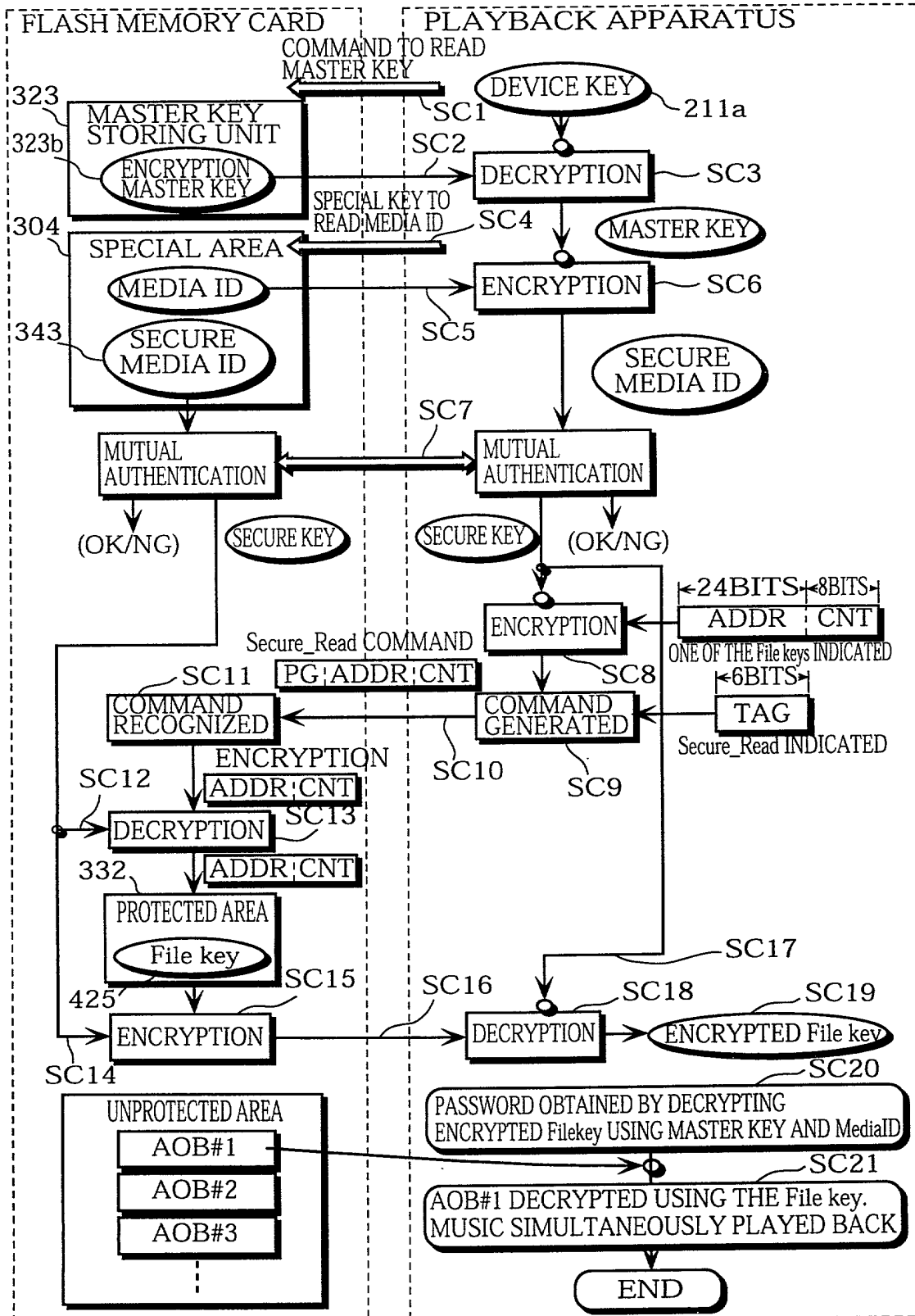




FIG. 70





# DECLARATION AND POWER OF ATTORNEY FOR U.S. PATENT APPLICATION

( ) Original    ( ) Supplemental    ( ) Substitute    ( ) PCT    ( ) Design

As a below named inventor, I hereby declare that: my residence, post office address and citizenship are as stated below next to my name; that I verily believe that I am the original, first and sole inventor (if only one name is listed below) or an original, first and joint inventor (if plural inventors are named below) of the subject matter which is claimed and for which a patent is sought on the invention entitled:

Title: A SEMICONDUCTOR MEMORY CARD, PLAYBACK APPARATUS, RECORDING APPARATUS,  
PLAYBACK METHOD, RECORDING METHOD, AND COMPUTER-READABLE RECORDING MEDIUM

of which is described and claimed in:

- (☒) the attached specification, or  
 ( ) the specification in the application Serial No. \_\_\_\_\_ filed \_\_\_\_\_;  
 and with amendments through \_\_\_\_\_ (if applicable), or  
 ( ) the specification in International Application No. PCT/ \_\_\_\_\_, filed \_\_\_\_\_, and as amended  
 on \_\_\_\_\_ (if applicable).

I hereby state that I have reviewed and understand the content of the above-identified specification, including the claims, as amended by any amendment(s) referred to above.

I acknowledge my duty to disclose to the Patent and Trademark Office all information known to me to be material to patentability as defined in Title 37, Code of Federal Regulations, §1.56.

I hereby claim priority benefits under Title 35, United States Code, §119 (and §172 if this application is for a Design) of any application(s) for patent or inventor's certificate listed below and have also identified below any application for patent or inventor's certificate having a filing date before that of the application on which priority is claimed:

COUNTRY	APPLICATION NO.	DATE OF FILING	PRIORITY CLAIMED
Japan	11-149893	May 28, 1999	Yes
Japan	11-236724	August 24, 1999	Yes
Japan	11-372606	December 28, 1999	Yes

I hereby claim the benefit under Title 35, United States Code, §120 of any United States application(s) listed below and, insofar as the subject matter of each of the claims of this application is not disclosed in the prior United States application in the manner provided by the first paragraph of Title 35, United States Code, §112, I acknowledge the duty to disclose information material to patentability as defined in Title 37, Code of Federal Regulations, §1.56 which occurred between the filing date of the prior application and the national or PCT international filing date of this application.

APPLICATION SERIAL NO.	U.S. FILING DATE	STATUS: PATENTED, PENDING, ABANDONED

And I hereby appoint John T. Miller, Reg. No. 21,120; Michael R. Davis, Reg. No. 25,134; Matthew M. Jacob, Reg. No. 25,154; Jeffrey Nolton, Reg. No. 25,408; Warren M. Cheek, Jr., Reg. No. 33,367; Nils E. Pedersen, Reg. No. 33,145 and Charles R. Watts, Reg. No. 33,142, who together constitute the firm of WENDEROTH, LIND & PONACK, L.L.P., attorneys to prosecute this application and to transact all business in the U.S. Patent and Trademark Office connected therewith.

I hereby authorize the U.S. attorneys named herein to accept and follow instructions from Nakajima Patent Attorneys Office as to any action to be taken in the U.S. Patent and Trademark Office regarding this application without direct communication between the U.S. attorneys and myself. In the event of a change in the persons from whom instructions may be taken, the U.S. attorneys named herein will be so notified by me.

Send Correspondence to

WENDEROTH, LIND & PONACK, L.L.P.  
2033 K Street, N.W., Suite 800  
Washington, DC 20006

Direct Telephone Calls to:

WENDEROTH, LIND & PONACK, L.L.P.  
Area Code (202) 721-8200

Direct Facsimile Messages to:

Area Code (202) 721-8250

<b>Full Name of First Inventor</b>	<b>FAMILY NAME</b> Hirota	<b>FIRST GIVEN NAME</b> Teruto	<b>SECOND GIVEN NAME</b>	
<b>Residence &amp; Citizenship</b>	<b>CITY</b> Moriguchi	<b>STATE OR COUNTRY</b> Japan	<b>COUNTRY OF CITIZENSHIP</b> Japan	
<b>Post Office Address</b>	<b>ADDRESS</b> 1-20-1-306, Kajimachi	<b>CITY</b> Moriguchi	<b>STATE OR COUNTRY</b> Japan	<b>ZIP CODE</b> 570-0015
<b>Full Name of Second Inventor</b>	<b>FAMILY NAME</b> Tagawa	<b>FIRST GIVEN NAME</b> Kenji	<b>SECOND GIVEN NAME</b>	
<b>Residence &amp; Citizenship</b>	<b>CITY</b> Katano	<b>STATE OR COUNTRY</b> Japan	<b>COUNTRY OF CITIZENSHIP</b> Japan	
<b>Post Office Address</b>	<b>ADDRESS</b> 5-5-305, Myoukenzaka	<b>CITY</b> Katano	<b>STATE OR COUNTRY</b> Japan	<b>ZIP CODE</b> 576-0021
<b>Full Name of Third Inventor</b>	<b>FAMILY NAME</b> Matsushima	<b>FIRST GIVEN NAME</b> Hideki	<b>SECOND GIVEN NAME</b>	
<b>Residence &amp; Citizenship</b>	<b>CITY</b> Studio City	<b>STATE OR COUNTRY</b> California, U.S.A.	<b>COUNTRY OF CITIZENSHIP</b> Japan	
<b>Post Office Address</b>	<b>ADDRESS</b> 10989 Bluffside Dr., #3217	<b>CITY</b> Studio City	<b>STATE OR COUNTRY</b> California	<b>ZIP CODE</b> 91604
<b>Full Name of Fourth Inventor</b>	<b>FAMILY NAME</b> Ishikawa	<b>FIRST GIVEN NAME</b> Tomokazu	<b>SECOND GIVEN NAME</b>	
<b>Residence &amp; Citizenship</b>	<b>CITY</b> Toyonaka	<b>STATE OR COUNTRY</b> Japan	<b>COUNTRY OF CITIZENSHIP</b> Japan	
<b>Post Office Address</b>	<b>ADDRESS</b> 4-6-14, Sanwacho	<b>CITY</b> Toyonaka	<b>STATE OR COUNTRY</b> Japan	<b>ZIP CODE</b> 561-0828

<b>Full Name of Fifth Inventor</b>	<b>FAMILY NAME</b> Inoue	<b>FIRST GIVEN NAME</b> Shinji	<b>SECOND GIVEN NAME</b>
<b>Residence &amp; Citizenship</b>	<b>CITY</b> Neyagawa	<b>STATE OR COUNTRY</b> Japan	<b>COUNTRY OF CITIZENSHIP</b> Japan
<b>Post Office Address</b>	<b>ADDRESS</b> 19-1-1142, Matsuya-cho	<b>CITY</b> Neyagawa	<b>STATE OR COUNTRY</b> Japan <b>ZIP CODE</b> 572-0086

<b>Full Name of Sixth Inventor</b>	<b>FAMILY NAME</b> Kozuka	<b>FIRST GIVEN NAME</b> Masayuki	<b>SECOND GIVEN NAME</b>
<b>Residence &amp; Citizenship</b>	<b>CITY</b> Arcadia	<b>STATE OR COUNTRY</b> California, U.S.A.	<b>COUNTRY OF CITIZENSHIP</b> Japan
<b>Post Office Address</b>	<b>ADDRESS</b> 501 Coyle Avenue,	<b>CITY</b> Arcadia	<b>STATE OR COUNTRY</b> California <b>ZIP CODE</b> 91008

<b>Full Name of Seventh Inventor</b>	<b>FAMILY NAME</b>	<b>FIRST GIVEN NAME</b>	<b>SECOND GIVEN NAME</b>
<b>Residence &amp; Citizenship</b>	<b>CITY</b>	<b>STATE OR COUNTRY</b>	<b>COUNTRY OF CITIZENSHIP</b>
<b>Post Office Address</b>	<b>ADDRESS</b>	<b>CITY</b>	<b>STATE OR COUNTRY</b> <b>ZIP CODE</b>

I further declare that all statements made herein of my own knowledge are true, and that all statements on information and belief are believed to be true; and further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code, and that such willful false statements may jeopardize the validity of the application or any patent issuing thereon.

1st Inventor Tenito Kinta Date May 8, 2000  
 2nd Inventor Henji Tagawa Date May 8, 2000  
 3rd Inventor Udeki Matsushita Date May 8, 2000  
 4th Inventor Tomokazu Ishikawa Date May 8, 2000  
 5th Inventor Shinji Inoue Date May 8, 2000  
 6th Inventor Masayuki Kozuka Date May 8, 2000  
 7th Inventor \_\_\_\_\_ Date \_\_\_\_\_

The above application may be more particularly identified as follows:

U.S. Application Serial No. \_\_\_\_\_ Filing Date \_\_\_\_\_  
 Applicant Reference Number \_\_\_\_\_ Atty Docket No. \_\_\_\_\_  
 Title of Invention \_\_\_\_\_